



# **Universidad Carlos III de Madrid**

## **Escuela Politécnica Superior**

**Ingeniería Técnica de Telecomunicación: Sistemas  
de Telecomunicación**

# **Videoconferencia docente desde cualquier lugar**

**Proyecto Fin de Carrera**

**Autora: María Luisa Calvo Frutos**

**Director: Dr. José Jesús García Rueda**

**Marzo, 2010**



**Departamento de Ingeniería Telemática**

**Escuela Politécnica Superior**

**Universidad Carlos III de Madrid**



**Proyecto Fin de Carrera**

**Ingeniería Técnica de Telecomunicación: Sistemas  
de Telecomunicación**

# **Videoconferencia docente desde cualquier lugar**

**Autora: María Luisa Calvo Frutos**

**Director: Dr. José Jesús García Rueda**

**Marzo, 2010**

# Agradecimientos

Muchas son las personas que me han apoyado a lo largo de esta difícil y larga etapa de mi vida en la que he estado dedicada por completo a la llamada “carrera”.

En primer lugar quería agradecer especialmente a mis padres, Conchi y Alfonso, la ayuda y apoyo que me han brindado durante todo este período universitario. Gracias por estar ahí siempre queriendo lo mejor para mí.

Por supuesto he de agradecer también especialmente el apoyo, compañía, comprensión, paciencia... que no ha dejado de ofrecerme Carlos, mi mejor amigo, consejero, compañero de viaje, mi referencia y el amor de mi vida. Gracias por hacer que mi vida tenga sentido.

Agradecer también especialmente el apoyo de mi hermana, Lorena. Siempre ha sido el ejemplo a seguir para mí. Gracias por enseñarme desde muy pronto a pelear por lo que se quiere y a moverme por la vida. Gracias también a mis tíos, M<sup>a</sup> Luisa y Andrés, a mis primos, Mamen y Andrés, y a mi abuela, “mami”.

Durante estos años, he conocido a mucha gente en la universidad que, aunque antes eran completamente desconocidos han sido los que en muchas ocasiones me han animado a no tirar la toalla antes de tiempo. Especialmente tengo que agradecer el apoyo de mis inseparables Pablo y Kiko. Pero también he de darle las gracias a Pedro, Sergio, Charly, Alberto, Jorge, Iván, María, Nacho, Goyo, Samu, Elena, Rober, Pablo, y un largo etcétera. Gracias a todos por hacerme reír sin parar durante todos estos años.

Por supuesto agradecer también a José Jesús, mi tutor del proyecto, y al departamento de Ingeniería Telemática, la oportunidad que me han dado de poder realizar el Proyecto Fin de Carrera con ellos.

Especialmente dedicado a mis abuelos Pepe y Alfonso.

# Resumen

El propósito de este proyecto es incorporar la videoconferencia como uno más de los métodos de enseñanza habituales, hacer más atractiva esta práctica para los alumnos y a su vez multiplicar sus posibilidades educativas aportando total movilidad a uno de los extremos de la comunicación. Será por tanto necesario romper con el esquema tradicional de videoconferencia que usan la mayor parte de instituciones actualmente en el que ambos extremos suelen estar localizados en salas especialmente adaptadas para llevar a cabo videoconferencias.

El grueso del proyecto se ha centrado en el diseño e implementación de una aplicación de videoconferencia propia llamada Teletrófono a través del paquete *Java Media Framework*. Por otro lado, un novedoso estándar del IEEE llamado WiMAX, ha sido la tecnología escogida para dar solución a una de las mayores dificultades que plantea el proyecto: dotar de acceso inalámbrico de banda ancha al extremo de la videoconferencia al que queremos dar total libertad en cuanto a su ubicación.

Se concluye que una tecnología a la que le ha costado despegar, la videoconferencia, unida a los novedosos estándares de transmisión inalámbrica que van apareciendo, puede dar lugar a soluciones no sólo aplicables en un entorno educativo, también aplicables a otros ámbitos tan interesantes como el de industria televisiva.

# Índice de contenido

---

	Pág.
<b>Capítulo I. INTRODUCCIÓN Y OBJETIVOS.....</b>	<b>1</b>
<b>Capítulo II. ESTADO DEL ARTE: LA VIDEOCONFERENCIA.....</b>	<b>7</b>
2.1.- Introducción.....	8
2.2.- Situación actual.....	11
2.3.- Precedentes.....	14
2.3.1.- Precedentes en diversos ámbitos.....	14
2.3.2.- Precedentes en educación.....	17
2.3.3.- Precedentes similares.....	22
2.4.- Programas de videoconferencia existentes.....	29
2.4.1.- Programas de videoconferencia en los que es necesaria instalación.....	29
2.4.2.- Programas de videoconferencia basados en web.....	31
<b>Capítulo III. ESTADO DEL ARTE: WiMAX.....</b>	<b>33</b>
3.1.- Introducción a WiMAX.....	34
3.2.- Estándares WiMAX.....	38
3.3.- Características generales WiMAX.....	39
3.4.- WiMAX frente a otras tecnologías.....	43
3.4.1. WiMAX vs. Wi-Fi.....	43
3.4.2. WiMAX vs. HSDPA.....	45
3.4.3. WiMAX vs. Mobile-Fi.....	47
3.5.- Licencias y operadores.....	49
3.5.1. Operadores de WiMAX en España.....	50
3.6.- Proyectos y casos prácticos WiMAX.....	52
3.6.1. Proyectos y casos prácticos en España.....	52
3.6.2. Otros casos prácticos.....	56
3.7.- Conclusiones.....	58
<b>Capítulo IV. DESARROLLO DE UN SOFTWARE DE VIDEOCONFERENCIA                 ESPECÍFICO.....</b>	<b>61</b>
4.1.- Requisitos del software de videoconferencia.....	62
4.2.- Software propio de Videoconferencia: Teletrófono.....	63

4.2.1.- Funcionalidades.....	64
4.2.2.- Diseño.....	68
4.2.3.- Implementación.....	95
4.3.- Problemas y Conclusiones.....	153
<b>Capítulo V. DISPOSITIVOS A UTILIZAR.....</b>	<b>163</b>
5.1.- Dispositivos en el extremo local de la videoconferencia.....	164
5.2.- Dispositivos en el extremo remoto de la videoconferencia.....	166
5.2.1.- Dispositivos móviles con tecnología WiMAX.....	167
5.2.2.- Elección de los dispositivos para el lado remoto.....	169
5.3.- Solución Final.....	172
<b>Capítulo VI. PRUEBAS REALIZADAS.....</b>	<b>175</b>
<b>Capítulo VII. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO.....</b>	<b>181</b>
7.1.- Conclusiones.....	182
7.2.- Futuras líneas de trabajo.....	184

## **BIBLIOGRAFÍA**

### **Anexo I. MANUAL TELETRÓFONO**

### **Anexo II. DIAGRAMAS DEL CÓDIGO**

# Índice de Figuras y Tablas

---

**Figura 2.1.-** Publicidad de 1968 que anunciaba la próxima llegada del “Picturephone”.

**Figura 3.1.-** Escenarios WiMAX.

**Figura 3.2.-** Tabla comparativa de estándares 802.16.

**Figura 3.3.-** Cuadro comparativo WiMAX vs. otras tecnologías inalámbricas.

**Figura 4.1.-** Secuencia de ejecución de un código fuente en Java.

**Figura 4.2.-** Escenario al que se da solución con “Teletrófono”.

**Figura 4.3.-** Esquema básico de videoconferencia a través de “Teletrófono”.

**Figura 4.4.-** Esquema de videoconferencia a través de “Teletrófono” con grabación de la sesión.

**Figura 4.5.-** JMF: Estados de un *Player*.

**Figura 4.6.-** JMF: Estados de un *Processor*.

**Figura 4.7.-** JMF: Transmisión RTP.

**Figura 4.8.-** JMF: Recepción RTP.

**Figura 4.9.-** JMF: Recepción y presentación de un flujo RTP.

**Figura 4.10.-** Diseño de la aplicación “Teletrófono”. Esquema ‘A’ lado remoto.

**Figura 4.11.-** Diseño de la aplicación “Teletrófono”. Esquema ‘B’ lado remoto.

**Figura 4.12.-** Diseño de la aplicación “Teletrófono”. Esquema ‘C’ lado remoto.

**Figura 4.13.-** Diseño de la aplicación “Teletrófono”. Esquema ‘D’ lado remoto.

**Figura 4.14.-** Diseño de la aplicación “Teletrófono”. Esquema ‘A’ lado local.

**Figura 4.15.-** Diseño de la aplicación “Teletrófono”. Esquema ‘B’ lado local.

**Figura 4.16.-** Diseño de la aplicación “Teletrófono”. Esquema ‘C’ lado local.

**Figura 4.17.-** “Teletrófono”: Esquema de puertos inicial.

**Figura 4.18.-** “Teletrófono”: Esquema de puertos definitivo.

**Figura 4.19.-** “Teletrófono”: Esquema definitivo Extremo Remoto.

**Figura 4.20.-** “Teletrófono”: Esquema definitivo Extremo Local.



**Figura 4.21.-** Reproductor de “Teletrófono”.

**Figura 4.22.-** Pantalla de presentación de “Teletrófono”.

**Figura 4.23.-** Ventana del menú principal de “Teletrófono”.

**Figura 4.24.-** Menú principal de “Teletrófono”: “Iniciar Videoconferencia desde el extremo Remoto”.

**Figura 4.25.-** Esquema de implementación de parte del Reproductor Teletrófono.

**Figura 4.26.-** Formato del archivo de texto con los subtítulos.

**Figura 4.27.-** Ventana del Reproductor “Teletrófono”.

**Figura 4.28.-** Ventana del Editor “Teletrófono”.

**Figura 4.29.-** Ventana del Editor “Teletrófono”, editando un subtítulo.

**Figura 4.30.-** Funcionamiento de la edición de subtítulos.

**Figura 4.31.-** Ventana de configuración de la videoconferencia desde el extremo Local.

**Figura 4.32.-** Ventana de configuración de la videoconferencia desde el extremo Remoto.

**Figura 4.33.-** Ventana de inicio de videoconferencia del extremo Local.

**Figura 4.34.-** Videoconferencia Teletrófono desde el lado Local.

**Figura 4.35.-** Inserción de subtítulo (I).

**Figura 4.36.-** Inserción de subtítulo (II).

**Figura 4.37.-** Clases en el lado local de la comunicación (I).

**Figura 4.38.-** Tabla con los formatos de vídeo soportados por JMF.

**Figura 4.39.-** Clases en el lado local de la comunicación (II).

**Figura 4.40.-** Ventana de inicio de videoconferencia del extremo remoto.

**Figura 4.41.-** Videoconferencia Teletrófono desde el lado Remoto.

**Figura 4.42.-** Clases en el lado Remoto de la comunicación.

**Figura 5.1.-** Instalación de Red en un Instituto de Educación Secundaria.

**Figura 5.2.-** “Asus Eee PC 1008HA”. Un *netbook* con WiMAX.

**Figura 5.3.-** “Samsung SPH-9200”. Un dispositivo ultra ligero con WiMAX.

**Figura 5.4.-** “SWT-P230”. Tarjeta PCMCIA para conexión WiMAX en 2,3 GHz.

**Figura 5.5.-** Dispositivos “MiMAX” de ‘Airspan’.

**Figura 6.1.-** Reproductor Teletrófono reproduciendo el vídeo de las pruebas grabado (con la herramienta de grabación Teletrófono) de la sesión de videoconferencia.

**Figura 6.2.-** Reproductor Teletrófono reproduciendo el vídeo de ejemplo grabado (con la herramienta de grabación Teletrófono) de la sesión de videoconferencia.

**Figura I.1.-** Menú principal de Teletrófono.

**Figura I.2.-** Menú principal de Teletrófono: “Iniciar Videoconferencia desde el extremo Remoto”.

**Figura I.3.-** Ventana de configuración de la videoconferencia desde el extremo local.

**Figura I.4.-** Ventana de inicio de videoconferencia del extremo local.

**Figura I.5.-** Videoconferencia Teletrófono desde el lado local.

**Figura I.6.-** Panel de inserción de subtítulos.

**Figura I.7.-** Introduciendo un subtítulo.

**Figura I.8.-** Ventana de videoconferencia desde el lado local con introducción de subtítulos.

**Figura I.9.-** Ventana de selección de la ruta y nombre del archivo de vídeo.

**Figura I.10.-** Mensaje que aparece al finalizar la grabación de la videoconferencia.

**Figura I.11.-** Ventana de configuración de videoconferencia desde el extremo remoto.

**Figura I.12.-** Ventana de inicio de videoconferencia del extremo remoto.

**Figura I.13.-** Videoconferencia Teletrófono desde el lado remoto.

**Figura I.14.-** Selector de archivos del Reproductor Teletrófono.

**Figura I.15.-** Ventana del Reproductor Teletrófono.

**Figura I.16.-** Diálogo que aparece al iniciar el Editor Teletrófono.

**Figura I.17.-** Ventana del Editor Teletrófono.

**Figura I.18.-** Panel de edición del Editor Teletrófono.

**Figura I.19.-** Ventana del Editor Teletrófono, editando un subtítulo.

**Figura I.20.-** Mensaje que indica que se ha modificado un subtítulo.



# CAPÍTULO I

## INTRODUCCIÓN Y OBJETIVOS

*“Empecé a darme cuenta de que en la vida diaria pasaba mucho tiempo conversando. Al comienzo ese conversar me parecía como un obstáculo para el trabajo 'real'. Pensaba que el trabajo real consistía en calcular, organizar, programar... Luego me di cuenta de que ese conversar era trabajo y que estas conversaciones tenían consecuencias. En aquel tiempo no tenía la noción del lenguaje como invención y constitución de la realidad; lo que yo llamo el papel ontológico del lenguaje vino después.”* [Fernando Flores Labra. Filósofo e ingeniero chileno].

El ser humano, tiene la necesidad de comunicarse, de ser escuchado y sobre todo de interactuar con los demás seres vivos que lo rodean. Por tanto, tal y como sugiere Fernando Flores, la comunicación es un proceso cotidiano, inherente al ser humano y más productivo de lo que en ocasiones pensamos.

Tornando hacia una definición de tipo académico podríamos definir la comunicación como: la transmisión de información de un lugar a otro, mediante un proceso en el cual se incluye un mensaje de un emisor a un receptor por medio de un canal.

En los últimos 150 años, y en especial en las dos últimas décadas, la reducción de los tiempos de transmisión de la información a distancia y de acceso a la información ha supuesto uno de los retos esenciales de nuestra sociedad, dando lugar a lo que se ha llamado: Revolución de la Información. Durante dicha revolución hemos sido testigos del rápido desarrollo de las computadoras, de las redes de ordenadores, del incremento en la capacidad y velocidad de los procesadores y del galopante incremento de la capacidad de almacenamiento electrónico. Estos desarrollos han sido una fuerza muy dinámica que ha afectado a la educación en general y especialmente a la educación a distancia, proporcionándole un nuevo, poderoso e interactivo medio para reducir las barreras de tiempo y espacio y poder llegar a sus objetivos.

La educación a distancia es una modalidad educativa en la que los estudiantes no necesitan estar físicamente presentes en el mismo lugar que el profesor. Históricamente educación a distancia significaba estudiar por correspondencia, pero en la actualidad se utilizan una gran variedad de medios electrónicos para llevar a cabo dicha empresa.

Si bien el desarrollo de las computadoras había permitido avances en su aplicación en la educación, no fue hasta que se reunieron los avances de las computadoras con los avances de las telecomunicaciones, cuando las aplicaciones a la educación se multiplicaron y expandieron de manera importante. La posibilidad de comunicar a las personas a través de estos exitosos aparatos ha abierto grandes e importantes oportunidades.

Cuando en un proceso de comunicación, caracterizado en su forma clásica por un emisor, un medio y un receptor, tanto el emisor como el receptor deben estar presentes de manera simultánea, se da un proceso síncrono de comunicación.

La videoconferencia es la tecnología de telecomunicaciones de uso síncrono más popular en la actualidad. Esta tecnología está convirtiendo a la educación a distancia en una alternativa viable para la educación tradicional en el aula de clase. Proporciona interacción, que otras formas de distribución no tienen, y permite retroalimentación instantánea, que les faltaba a los intentos iniciales de la educación a distancia. En la videoconferencia se envía audio y vídeo. El aprendizaje a través de videoconferencia facilita la formación a distancia e incrementa su eficacia, ya que si el mensaje se recibe a través de la vista y el oído, éste se recordará más fácilmente.

Una de las preguntas claves asociadas con la tecnología educativa es si contribuye o no al aprendizaje de los estudiantes. En los numerosos estudios que se han llevado a cabo sobre el tema, se han utilizado dos enfoques principales para investigar a los medios de comunicación:

- En el primer enfoque se compara el medio nuevo (ordenador), con un aula tradicional. Algunos estudios han encontrado rendimientos más altos de los estudiantes cuando se utilizaron programas interactivos de computadoras, incluyendo correo electrónico, video de un sentido o dos y multimedia.
- Otros estudios han seguido otro enfoque, centrándose en el contexto de aprendizaje más que en algún medio específico de envío. Estos estudios han mostrado que los estudiantes obtienen mejores resultados cuando se combinan varios medios de envío y técnicas de enseñanza. Han estudiado los efectos del uso de la videoconferencia de escritorio utilizada para evaluar el trabajo de otros compañeros o el efecto de participar en grupos de trabajo amplios. La interacción entre grupos de trabajo y las nuevas tecnologías educativas generalmente producen resultados positivos en los estudiantes.

El principal objetivo de este proyecto es introducir los beneficios de las tecnologías que se utilizan habitualmente en la educación a distancia, en un entorno tradicional de enseñanza. El proyecto pretende a través de su solución facilitar la introducción de tecnologías de comunicación síncronas, en concreto de la videoconferencia, en un escenario específico como el que se detalla a continuación.

Situémonos en un centro educativo, por ejemplo un instituto de educación secundaria. Supongamos que un profesor que habitualmente da clase en ese instituto se encuentra hoy en un escenario remoto. Este profesor desea mostrar a sus alumnos, que no se encuentran con él, ese escenario para explicarles más detalladamente algunos temas que han visto últimamente en clase y que están íntimamente relacionados con el lugar en el

que él se encuentra. Dicho entorno remoto puede ser cualquier lugar que se nos ocurra, tanto lugares en el interior de un edificio, casa, museo... como lugares al aire libre como puede ser una granja, un bosque, unas ruinas romanas o la calle de cualquier ciudad o pueblo. Sin embargo, los alumnos se encuentran a kilómetros de distancia, concretamente en el instituto. El deseo de este profesor es transmitir información audiovisual en directo a un aula donde se encuentran sus educandos, acompañados normalmente de otro profesor.

En esta situación el grueso de la información será transmitido desde el lado remoto en el que se encuentra el profesor hacia el aula, pero también los alumnos deberán poder transmitirle de algún modo sus dudas y cuestiones a este profesor. No debemos olvidar que la retroalimentación, preferiblemente inmediata, es un aspecto fundamental en la educación. El profesor que se encuentra en el aula con los alumnos podría actuar como moderador transmitiendo él mismo las dudas de los alumnos a su compañero en remoto e incluso, si el programa utilizado lo permitiese, podría encargarse de ir introduciendo rótulos durante el evento bajo la imagen del profesor a modo informativo al más puro estilo de un riguroso directo televisivo.

Asimismo sería muy interesante que se pudiese aprovechar la experiencia una vez que la sesión de videoconferencia entre el profesor en remoto y los alumnos haya finalizado. Sería por tanto deseable, que toda la información audiovisual transmitida durante el evento pudiese ser grabada y almacenada. De este modo, una vez finalizada la videoconferencia, el profesor podría utilizar la grabación para realizar otras actividades relacionadas, que se dejan a la imaginación de éste, aprovechando así la “visita virtual” realizada por los alumnos.

La necesidad que nos surge en este proyecto de que la aplicación de videoconferencia lleve a cabo determinadas tareas, aparte de la transmisión de audio y vídeo, nos llevará a desarrollar un *software* propio de videoconferencia que se llamará Teletrófono. Para la implementación de ese *software* se ha decidido usar ‘Java’ como lenguaje de programación y, más concretamente, su paquete JMF (*Java Media Framework*). El principal objetivo de desarrollar el programa de videoconferencia a través de JMF, es poder evaluar la idoneidad de este paquete a la hora de implementar aplicaciones de videoconferencia de tipo educativo que requieren algo más que la transmisión por la red de flujos multimedia.

Este proyecto propone buscar una solución fiable, económica y sencilla, de modo que no requiera un gran número de personas dedicadas a la tarea ni una especialización en la materia por parte de los actores.

De un tiempo a esta parte la videoconferencia se ha estado utilizando en muy diferentes entornos: reuniones, congresos, educación a distancia, medicina, justicia... Sin embargo en la mayoría de los entornos citados hay un diseño y uso prácticamente idéntico del esquema de videoconferencia. En ambos extremos se dispone de una conexión vía RDSI o ADSL a través de un módem conectado a la línea telefónica. El ordenador se conecta a través de cable a dicha red y la videoconferencia tiene lugar en una sala equipada a todo detalle para tal efecto.

Pero la situación planteada en este proyecto rompe en algunos aspectos con este esquema: aunque en un extremo de la comunicación (el aula del centro de enseñanza) tendremos probablemente un acceso a Internet tradicional, como los que hemos nombrado en el párrafo anterior, no sucede lo mismo en el extremo remoto donde se encuentra el profesor en solitario (que como hemos dicho puede ser cualquier lugar que imaginemos).

Al romper con el esquema tradicional, pensado para obtener la mayor calidad posible, hay que ser conscientes de que perderemos parte de esa calidad, pero ganaremos en otros aspectos. Ganaremos en movilidad, lo que permite extender la videoconferencia más allá de las fronteras actuales. Ganaremos en comodidad y también probablemente obtengamos una solución más económica y, por tanto, más accesible.

Para conseguir la deseada movilidad necesitaremos de una comunicación inalámbrica, que contacte entre sí ambas partes a través de ondas electromagnéticas que viajen por el aire evitando así la necesidad de tender cables de extremo a extremo. La tendencia a la movilidad y la ubicuidad hacen que cada año aumente la demanda de sistemas inalámbricos por parte de los usuarios. Esto ha hecho que en los últimos años se desarrollasen multitud de tecnologías capaces de transmitir información sin la necesidad de cables. Una de las tecnologías más novedosas en este ámbito se denomina WiMAX (*Worldwide Interoperability for Microwave Access*) y es la tecnología elegida en este proyecto para dar solución al problema de la transmisión de audio y vídeo entre un entorno remoto desconocido y un entorno local bien definido.

En el siguiente capítulo, **capítulo II** se profundizará en cuanto a la evolución de la videoconferencia y los proyectos que se han llevado a cabo con esta tecnología en diversos ámbitos.

Para poder situar al protagonista principal de nuestra videoconferencia en un entorno remoto indeterminado necesitamos determinar una solución única para acceder a Internet, con la que podamos disfrutar de total movilidad. En el **capítulo III** estará dedicado, por tanto, a la tecnología elegida para acceder inalámbricamente a la red sobre la que se va a llevar a cabo la videoconferencia: WiMAX.

Durante el **capítulo IV** nos centraremos exclusivamente en el programa de videoconferencia a través del cual se comunicarán los dos extremos. Definiremos los requisitos que guiarán al diseño de la aplicación y se citarán algunos de los programas de videoconferencia existentes en el mercado. Explicaremos los conceptos básicos del lenguaje escogido para desarrollar la aplicación para pasar después a explicar detalladamente cómo se ha llevado a cabo el diseño y la implementación de Teletrófono, nuestro programa de videoconferencia.

Una vez tenemos listo el programa de videoconferencia, debemos dejar completamente definida la solución, por lo que a lo largo del **capítulo V** estudiaremos los dispositivos que necesitamos utilizar en cada uno de los extremos de la comunicación, sin perder de vista los principales objetivos del proyecto.

Una de las últimas etapas de la realización del proyecto consistirá en el montaje y puesta en marcha de la solución ideada en un entorno real. Las pruebas realizadas se reflejarán en el **capítulo VI** de este documento.

En los **capítulos VII y VIII** se expondrán las conclusiones a las que se ha llegado tras la realización del proyecto, así como las posibles futuras líneas de trabajo que se pueden llevar a cabo a partir de este proyecto.

## OBJETIVOS DEL PROYECTO

El objetivo del proyecto es claro: comunicar mediante videoconferencia dos extremos físicamente alejados entre sí, en un contexto educativo en el que uno de los extremos debe gozar de una total movilidad. Se usará la tecnología WiMAX como modo de acceso a Internet desde uno de los extremos, así como se desarrollará un programa de videoconferencia a través del lenguaje de programación 'Java'.

Como acabamos de ver, en este proyecto se plantea un escenario muy concreto en el que: en el extremo al que llamamos 'local' encontramos alumnos tutelados por un profesor en un aula de un centro educativo, y en el extremo al que llamamos 'remoto' encontramos un profesor situado en un entorno cualquiera desde el cual le interesa dar clase. Dicho escenario remoto puede ser cualquier lugar que se nos ocurra, tanto lugares interiores como exteriores, una sala de un museo o el jardín botánico de la ciudad. Mientras en el aula disponemos de los dispositivos y comunicaciones básicas para poder llevar a cabo una videoconferencia de modo tradicional, en la ubicación remota no tenemos en principio ninguna facilidad para realizar la videoconferencia. Además, la solución deberá ser única sea cual sea el lugar geográfico elegido para ubicar el extremo remoto.

Tras el planteamiento general del proyecto, vamos a citar los aspectos que se deben estudiar a fondo. Lo que parece claro es que será el extremo remoto el que nos imponga los límites de nuestro proyecto, pues es allí donde encontramos la mayoría de las dificultades a la hora de llevar a cabo la videoconferencia. Comencemos entonces a plantear objetivos concretos.

- Moviéndonos dentro de las limitaciones que nos impone la situación, trataremos de conseguir una solución que nos proporcione una buena calidad. No sólo se buscará la calidad en la comunicación entre los extremos, sino también la calidad referida al entorno virtual en el que se transmiten los conocimientos, es decir, que el *software* de videoconferencia que utilicemos nos proporcione más posibilidades aparte de la mera transmisión de la voz e imagen del profesor.

En la situación planteada, queremos que un profesor en un entorno remoto sea capaz de transmitir a los alumnos que se encuentran en el aula, conocimientos relacionados con el lugar en el que él se encuentra. Será por tanto imprescindible que desde el extremo remoto se envíe información audiovisual al extremo local. Aunque la mayor parte de la información será transmitida desde el extremo remoto, es imprescindible, en un entorno educativo, dotar a los alumnos de retroalimentación inmediata. Será deseable, por tanto, que el software nos ofrezca la posibilidad de que los alumnos puedan, no sólo ver y oír al profesor, sino también formularle preguntas en tiempo real.

Sin embargo, nos gustaría que el programa de videoconferencia también permitiese realizar otras tareas. Por ejemplo, el profesor situado en el aula, podría ir insertando rótulos bajo la imagen recibida desde remoto a modo de directo televisivo, haciendo así la experiencia más atractiva para los alumnos. Esta herramienta de inserción de subtítulos también podría ser útil en el caso en el que participaran en la experiencia alumnos con deficiencias auditivas.

Otro objetivo del proyecto es que la experiencia no finalice cuando finaliza la sesión de videoconferencia, por ello es un requisito imprescindible del *software* de videoconferencia que nos permita grabar la sesión. Los rótulos, de los que hemos



hablado en el párrafo anterior, podrían quedar grabados también para que así sirvan de guía en futuras visualizaciones de la sesión. Se desea de esta forma, que tras la experiencia en directo, los alumnos puedan volver a visualizar la grabación de la videoconferencia cuando deseen, surgiendo así múltiples opciones adicionales a imaginación del profesor para aprovechar la experiencia. Se pretende de este modo que la videoconferencia, usada como herramienta educativa, resulte todavía más atractiva a profesores y alumnos.

Siguiendo la política del proyecto, el *software* deberá ser extremadamente sencillo de utilizar, para que los profesores en los extremos sean capaces de defenderse sin problema sin la ayuda de ningún técnico a pesar de que no sean expertos en sistemas de videoconferencia.

Dada la lista de requisitos que pedimos al software de videoconferencia será indispensable que nosotros mismos desarrollemos la aplicación, ajustándonos así lo más fielmente posible a cada uno de los objetivos del proyecto.

- Para desarrollar el programa de videoconferencia se ha elegido Java como lenguaje, y en concreto su paquete que trata con datos multimedia, llamado JMF (*Java Media Framework*). Otro de los objetivos de proyecto es, por tanto, probar la capacidad de JMF para el desarrollo de una aplicación de videoconferencia educativa que requiere algo más que la transmisión de voz y datos de un extremo a otro.

- Uno de los mayores problemas a los que nos deberemos enfrentar será encontrar una solución para conectar el extremo remoto a una red de comunicación (en concreto, a la misma red a la que esté conectado el instituto), dondequiera que este extremo esté. Se ha decidido resolver esta situación a través de una tecnología de acceso inalámbrico de banda ancha llamada WiMAX. Será entonces otro objetivo del proyecto investigar y estudiar las características y situación concreta de esta tecnología, analizando sus ventajas e inconvenientes para deducir de este modo qué beneficios puede aportar a nuestro proyecto.

- Deberemos buscar una solución de *hardware* para el extremo remoto que sea fácil de transportar. Se tratará de minimizar el número de dispositivos necesarios en ese entorno para proporcionar comodidad al participante allí situado. La solución en este extremo deberá asimismo ser muy sencilla en el manejo, de modo que no necesitemos que el profesor vaya acompañado de ningún técnico.

- Trataremos de conseguir una solución que, en su conjunto, sea económica, de modo que resulte accesible en este aspecto para cualquier institución, centro educativo o empresa que la desee implementar. Este es un aspecto clave en nuestro proyecto ya que se pretende que la videoconferencia se introduzca en las aulas habitualmente como una técnica más de enseñanza.

Debido a los dos puntos anteriores, deberemos prescindir de la gran mayoría de dispositivos y elementos que se usan habitualmente en videoconferencias orientadas a la educación; normalmente se trata de dispositivos caros que a cambio proporcionan una excelente calidad.

## CAPÍTULO II

### ESTADO DEL ARTE: LA VIDEOCONFERENCIA

*“Francis Bennet se había despertado aquella mañana de muy mal humor. Hacía ocho días que su esposa estaba en Francia. Se encontraba, pues, algo afligido. Aunque parezca increíble, en los diez años de matrimonio ésta era la primera vez que Mrs. Edith Bennet, modelo profesional, se ausentaba de casa tanto tiempo. Habitualmente, dos o tres días bastaban para sus frecuentes viajes a Europa, y particularmente a París, donde iba a comprarse sombreros. La primera preocupación de Francis Bennet fue poner en funcionamiento su fonotelefoto, cuyos hilos comunicaban con su mansión en los Campos Elíseos. El teléfono complementado por el telefoto, otro gran logro de la ciencia moderna. Si desde hace tantos años se transmite la palabra mediante corrientes eléctricas, la transmisión de imágenes por medio de espejos sensibles conectados mediante cables es una cosa apenas de ayer. Valioso descubrimiento. Esta mañana Mr. Bennet pensó en su bendito inventor cuando percibió perfectamente a su mujer, reproducida en un espejo telefónico a pesar de la enorme distancia que los separaba.”*  
[Julio Verne, 1888. *La jornada de un periodista americano en el 2890*].

A todos sigue asombrando el poder visionario de Julio Verne y, más aún, cuando somos conscientes de que ha pasado más de un siglo desde que escribió sus novelas. En el cuento del que se han extraído estas líneas aparece la primera referencia a un videoteléfono que se conoce.

A unos les parecerá mucho, a otros les parecerá poco, pero el caso es que para trasladar la ciencia ficción a la realidad ha tenido que pasar ni más ni menos que un siglo. En este capítulo se hará una breve introducción a lo que hoy conocemos como videoconferencia comenzando con una mirada retrospectiva a su evolución. Trataremos de recopilar todos los entornos en los que esta tecnología está siendo utilizada en la actualidad y se estudiarán diferentes ejemplos, habiendo buscado con más empeño aquellos casos que más características pueden tener en común con este proyecto.

## 2.1. INTRODUCCIÓN

La idea de las videocomunicaciones personales, los videoteléfonos, es, al margen de la ciencia ficción, antigua. En el libro de John R. Pierce y A. Michael Noll llamado “*Señales. La Ciencia de las Telecomunicaciones*”, podemos leer algo de la historia de estas tecnologías. Allí se nos cuenta que en la Exposición Universal de Chicago, a principios de los años 30, antes de que existieran las cadenas de televisión, se expuso un sistema de videotelefonía. Un cable coaxial enlazó las centrales públicas de videoteléfonos de cuatro ciudades alemanas desde 1935 hasta 1938. El interés en la comunicación utilizando video creció con la disponibilidad de la televisión comercial iniciada en 1940. AT&T (*American Telephone and Telegraph*), la histórica compañía estadounidense fundada en 1885, mostró su primer ‘Picturephone’ de uso doméstico en la Feria Mundial de 1964 de Nueva York. La demostración consistía en comunicar a través del novedoso aparato a los visitantes de la Feria Mundial de Nueva York con los visitantes de ‘*Disneyland*’ en California. Se trataba de un prototipo de videoteléfono que requería de líneas de comunicación bastante costosas para transmitir vídeo en movimiento. La primera versión del ‘Picturephone’ podía transmitir una imagen en blanco y negro cada dos segundos. Ese mismo año, la compañía inició una comercialización limitada del servicio, también entre centros públicos de tres grandes ciudades. Después de varios ensayos dirigidos a potenciar el servicio de cara a las empresas de negocios, AT&T introdujo su ‘Picturephone’ en Pittsburgh en 1970, y en Chicago en 1971. El servicio costaba 160 dólares por los primeros 30 minutos al mes y 25 centavos por cada minuto añadido. En 1971, un centro de estudios estratégicos y prospectivos de la costa Oeste de EEUU predijo un despliegue de 2 millones de ‘Picturephones’ en uso para 1985. ¡Nada más lejos de la realidad!

La verdad es que el ‘Picturephone’ presentaba varios problemas. El primero de ellos, era el precio. Otro problema era que aunque para una llamada a través de un teléfono tradicional se necesitaba solamente un par de cobre, para cada llamada local, el ‘Picturephone’ necesitaba tres pares de cobre (dos para el vídeo y uno para el audio). En largas distancias, un solo cable soportaba varias llamadas, así que no se disponía de suficiente ancho de banda para que operase un ‘Picturephone’ nada más que en las llamadas dentro de una misma ciudad. Los ‘Laboratorios Bell’, estaban por entonces estudiando cómo comprimir la señal de vídeo.

Los servicios de video-teleconferencia de AT&T que empezaron a principios de los setenta, significaban una alternativa lógica a los viajes de negocios hechos simplemente para acudir a reuniones. El servicio estaba disponible en unas salas especialmente equipadas en cinco de las principales ciudades de Estados Unidos. No obstante su utilización fue baja.

A lo largo de los años 70 se realizaron progresos sustanciales en muchas áreas clave, los diferentes proveedores de redes telefónicas empezaron una transición hacia métodos de transmisión digitales. La industria de los ordenadores también avanzó enormemente tanto en la capacidad como en la velocidad de procesamiento de datos y se descubrieron y mejoraron significativamente los métodos de muestreo y conversión de señales analógicas (como las de audio y vídeo) en digitales.



Figura 2.1. Publicidad de 1968 que anunciaba la próxima llegada del 'Picturephone'.

La necesidad de una compresión fiable, imprescindible en vídeo digital, de datos digitales fue crítica, apareciendo a principios de los 80 algunos métodos de compresión, fueron llamados "vídeo *codecs*". Los *codecs* de principios de los 80's utilizaron una tecnología conocida como codificación de la Transformada Discreta del Coseno (DCT). Usando esta tecnología DCT las imágenes de video pueden ser analizadas para encontrar redundancia espacial y temporal. La redundancia espacial es aquella que puede ser encontrada dentro de un cuadro sencillo de video, "áreas de la imagen que se parecen bastante que pueden ser representadas con una misma secuencia". La redundancia temporal es aquella que puede ser encontrada de un cuadro de la imagen a otro " áreas de la imagen que no cambian en cuadros sucesivos". Combinando todos los métodos mencionados anteriormente, se logró obtener una razón de compresión de 60:1. A mediados de los 80's se observó una mejora dramática en la tecnología empleada en los *codecs* de manera similar, se observó una baja substancial en los costos de las medios de transmisión. El precio de los *codecs* cayó casi tan rápido como aumentaron los porcentajes de compresión. En 1990 los *codecs* existentes en el mercado habían reducido su costo en más de un 80%, añadiendo a esto una reducción en el tamaño.

Los *codecs* para videoconferencia actuales emplean una razón de compresión hasta de 1600:1 (56 Kbps), consiguiendo un costo del uso de la red telefónica aproximado al de una llamada telefónica. Pero utilizar razones de compresión tan grandes tiene como

desventaja la degradación en la calidad y en la definición de la imagen. (Fuente: Chacón, 2003)

El auténtico despliegue de los sistemas de videoconferencia comenzó a partir de 1996, cuando la Unión Internacional de Telecomunicaciones (UIT) inició la elaboración de normas para la codificación de la videoconferencia. Comenzó con el estándar H.263 para reducir el ancho de banda para la transmisión de baja velocidad binaria y H.323 para la comunicación de paquetes basada en las comunicaciones multimedia.

En 1999, el *Moving Picture Experts Group* desarrolló MPEG-4 como una norma ISO (*International Organization for Standardization*) para contenidos multimedia. Esto condujo al estándar H.264, también conocido como AVC (Advanced Video Codec), que se utiliza para comprimir las señales de alta definición sobre IP (*Internet Protocol*).

La mayoría de los nuevos productos de videoconferencia ahora incluyen H.264, así como capacidades H.263.

## 2.2. SITUACIÓN ACTUAL

Como hemos visto en el apartado anterior, la videoconferencia no es una nueva tecnología que haya surgido de la noche a la mañana, ni mucho menos recientemente. Lo que sí es cierto es que su uso popular, al margen de las grandes corporaciones, no comenzó hasta hace muy pocos años. Han hecho falta 35 años para que las tecnologías evolucionasen lo suficiente como para proporcionar una videoconferencia de calidad y asequible para una sociedad que ha cambiado sustancialmente.

En 1968, en pleno apogeo del movimiento contracultural, Theodore Roszak expresaba sus ideas sobre el papel de la ciencia y la tecnología en el mundo contemporáneo: *«Cualesquiera que sean las demostraciones y los beneficiosos adelantos que la explosión universal de la investigación produce en nuestro tiempo, el principal interés de quienes financian pródigamente esa investigación seguirá polarizado en el armamento, las técnicas de control social, la mercancía comercial, la manipulación del mercado y la subversión del proceso democrático a través del monopolio de la información y del consenso prefabricado»*.

Las palabras de Roszak, tremendas y exageradas como corresponden a un teórico de la contracultura, reflejan, no obstante, el espíritu de los tiempos que corrían: una creciente sensibilidad social y una preocupación política por las consecuencias negativas de una ciencia y una tecnología fuera de control. Es lo que se ha llamado «síndrome de Frankenstein», que empezó a extenderse en la opinión pública de los años 60 y 70.

Por el contrario en estos primeros años del siglo XXI el sustento tecnológico, la cultura de la sociedad global, la cibercultura, en el escenario de la postmodernidad, ha generado profundos cambios sociales, económicos y culturales, un nuevo tipo de sociedad, una sociedad inmersa en el vertiginoso desarrollo tecnológico, en el snobismo de la última tecnología.

Actualmente, las grandes consultoras de tecnologías de la información prevén que el incremento de la incidencia de las tecnologías de videoconferencia será de un 30% anual en los próximos 3 años. Hace relativamente poco tiempo, los desplazamientos tal y como los entendemos hoy no existían. Emprender un viaje suponía un elevado coste económico o una inversión importante de tiempo. Los medios de transporte en los últimos 50 años han sufrido una transformación incuestionable. Si a este abaratamiento de coste y tiempo en movernos, le añadimos las necesidades surgidas de un mercado globalizado, tenemos la respuesta a los flujos humanos que circulan por el mundo. En las últimas décadas muchos trabajadores han tenido que trasladarse, por motivos laborales, para reuniones de dos horas, a ciudades situadas a grandes distancias. Ciertamente es que hay empleos en los que estar presente es una necesidad, sin embargo otras muchas veces con nuestra presencia virtual se soluciona el problema. Los desplazamientos implican necesariamente un coste económico y temporal, pudiendo incidir negativamente en la productividad, equilibrio trabajo-vida privada e incluso en el medioambiente. (Fuente: Márquez, 2009)

La videoconferencia ya no es una tecnología cara y exclusiva de grandes instalaciones, pero la variedad de equipos existentes y las diferencias en coste y complejidad de uso, nos obliga a saber elegir el adecuado a nuestras necesidades, tanto en cuanto a equipo, como en lo referente a la conexión. Cada día aumentan las alternativas de comunicación

audiovisuales, podemos llevar a cabo este tipo de comunicación a través de la web o incluso a través de teléfonos celulares equipados.

Las ventajas de las videoconferencias de hoy en día son muchas. Para empezar, ahorramos el tiempo de viaje. En las grandes ciudades, moverse entraña dificultades. Coordinarse con los medios, encontrar aparcamiento, la lluvia, el tráfico, son circunstancias que nos hacen perder tiempo, un bien de incalculable valor en nuestras apresuradas jornadas. Por otro lado, el uso y abuso de los medios de transporte vienen provocando una alta emisión de gases contaminantes. Reducir nuestros desplazamientos a lo estrictamente necesario, ayuda en gran medida a preservar el medio ambiente. Y como tercera ventaja, con el sistema de videoconferencias no sólo reducimos tiempo y emisiones contaminantes, también reducimos los costes. Muchas empresas empiezan a plantearse la utilidad de sus encuentros físicos. Hasta el momento habíamos tratado aspectos que nos interesan a nivel de individuo, sin embargo estos gastos en desplazamientos afectan de un modo directo a los presupuestos tanto de las grandes multinacionales como de las pequeñas o medianas empresas. (Fuente: Márquez, 2009)

En un estudio encargado por ‘Tandberg’ a la firma ‘Ipsos Mori’ en 2003 (que podemos consultar en la página web

[http://www.tandberg.com/collateral/tandberg\\_videoconfering\\_travel\\_survey.pdf](http://www.tandberg.com/collateral/tandberg_videoconfering_travel_survey.pdf)) se pone de manifiesto que una de cada tres reuniones que se producen en los viajes de negocios podría realizarse a través de videoconferencias. Además se comenta que este tipo de soluciones pueden evitar el estrés derivado de los viajes, que afecta al 51% de los hombres y mujeres de negocio españoles y hasta al 81% de los italianos. El estudio afirma que incluso un 7% de los ejecutivos europeos estaría dispuesto a cambiar de trabajo por viajar con menor frecuencia, así como dice que un 15% de los directivos europeos declara ser menos productivo cuando tiene que viajar para celebrar una reunión. Según otro estudio de otra consultora, ‘Quocirca’ (al que podemos acceder a través de su página web <http://www.quocirca.com>), un 55% de los hombres y mujeres de negocios europeos admiten que sus viajes de negocio son innecesarios y un 29% señala la videoconferencia como la mejor alternativa para realizar reuniones productivas.

- Actualmente la mayoría de las grandes compañías o instituciones utilizan las videoconferencias para:

Reuniones a distancia: Como se ha comentado, en muchos casos ya no es necesario llevar a cabo largos viajes simplemente para asistir a una reunión.

Telemedicina: En aplicaciones médicas se puede desde intercambiar opiniones con colegas hasta dirigir complejas cirugías sin necesidad de estar presente en un pabellón. Con video de alta calidad se pueden ver hasta los más mínimos detalles de una operación, y con la comunicación en tiempo real, ir siguiendo o dirigiendo cada paso. El diagnóstico clínico remoto por videoconferencia se usa más frecuentemente en áreas rurales.

Conferencias: Ya no es necesario estar físicamente presente para dictar conferencias y dirigirse a un público numeroso. Se puede participar en conferencias a distancia sin necesidad de estar presente en el auditorio y disfrutando de la posibilidad de interactuar en tiempo real.

Educación: La educación es una herramienta que permite el desarrollo de la sociedad, es por ello que las instituciones buscan cada vez más opciones que les permitan llegar a

más gente. Actualmente se pueden realizar cursos a distancia o hacer clases más interactivas gracias a la inclusión de la tecnología de videoconferencia.

Telejusticia: La llegada de la videoconferencia a los tribunales de justicia ayuda también a la protección de testigos o a la efectividad de las presentaciones en juicios y otros. Se puede tener a personas en salas completamente separadas y hacerlos interactuar con el resto, contar sus testimonios o presentar sus pruebas sin necesidad de que tengan que estar en la misma habitación que sus contrapartes.

Servicio al cliente: Obtener soporte inmediato en productos o servicios. A las empresas les permite dar soporte y ayuda a clientes sin tener que enviarles un técnico o representante. Si vendes un producto o servicio que necesita un soporte, darle este servicio por video conferencia reducirá costes y mejorará las relaciones con el cliente al darle una asistencia rápida y efectiva.

A la videoconferencia también se le da actualmente otros muchos usos como: control de la manufactura, contratación/entrevistas, supervisión, adiestramiento/capacitación, etc.



## 2.3. PRECEDENTES

En este apartado se van a exponer, en primer lugar, algunos casos prácticos sobre usos que se han hecho o se hacen de la videoconferencia en algunos de los campos que se han mencionado en el apartado anterior. En segundo lugar, nos aproximaremos a casos que pueden estar más relacionados con este proyecto. Se mencionarán, por tanto, casos concretos sobre el uso de la videoconferencia en educación. Por último, se han escogido cuatro casos concretos de pruebas o aplicaciones con videoconferencia que nos interesan especialmente debido a la similitud que presentan con este proyecto.

### 2.3.1. PRECEDENTES EN DIVERSOS ÁMBITOS

▪ **La guerra del Golfo** en 1991 introdujo a algunas corporaciones internacionales a valorar la videoconferencia cuando el viaje es difícil o peligroso. Algunos ejecutivos utilizaron sistemas de videoconferencia para manejar operaciones transnacionales durante la guerra. Con esta Guerra se dieron a conocer al mundo las potencialidades de la videoconferencia, al facilitar la interacción de los ingenieros de petróleo en la zona del Golfo Pérsico con sus casas matrices en los EE.UU. En esta época los equipos que se vendían eran ya más ligeros y reducidos. En el caso de medicina de emergencia el telediagnóstico puede emplearse para tomar una decisión crítica sobre si evacuar o no el paciente hacia un hospital central. Esta modalidad de telediagnóstico se implementó exitosamente durante la Guerra mediante la transmisión de imágenes de tomografía computada a través de un sistema satelital de tele radiología para determinar si un soldado podía ser tratado en el campo de batalla o debía ser evacuado. Durante la Guerra del Golfo, el uso de tecnologías de voz y videoconferencia se incrementó, en menos de una semana, entre un 25% y un 30%. (Más información: [http://www.bioingenieros.com/empresas/historial/revista\\_9.txt](http://www.bioingenieros.com/empresas/historial/revista_9.txt))

▪ **El primer juicio por videoconferencia** en España se celebró el 7 de Mayo de 2001 desde la Universidad de Islas Baleares. La Audiencia de Sevilla pudo escuchar en tiempo real las respuestas al interrogatorio a tres testigos y tres peritos como si realmente se encontraran en la misma sala de vistas. El delito se había cometido en Sevilla, pero la denuncia había sido puesta en Mallorca, donde residía la víctima. Gracias al sistema de videoconferencia los testigos y peritos que viven en Mallorca no tuvieron que viajar para declarar. La UIB cedió el uso de una de sus dos salas de videoconferencias para la experiencia. En la sala de la UIB había un monitor de televisión en el que se veía al tribunal sevillano y a las personas que realizaban las preguntas, mientras que se grababan simultáneamente con una cámara las imágenes y el sonido de los que declaraban y se enviaban a la Audiencia andaluza. (Más información: <http://www.uib.es/premsa/maig01/dia-08/332519.htm> ).

▪ **La red OTN** (*Ontario Telemedicine Network*) en Canadá, se creó en Abril de 2006 mediante la fusión de las tres redes de telemedicina provinciales que existían hasta entonces: *CareConnect* (Ontario Este), Red NORTE (centro y norte de Ontario) y *VideoCare* (suroeste de Ontario). Se trata de un amplio programa de telesalud basado en H.323 para hospitales remotos y clínicas en las zonas más al norte de la citada provincia. Se emplea una red privada IP (dedicada a aplicaciones de salud) para enlazar más de 60 sitios en el norte hacia los grandes hospitales y centros de enseñanza. Actualmente facilitan cientos de consultas por videoconferencia cada semana así como proporciona un muy amplio programa educativo empleando las mismas tecnologías. En 2008, en Ontario se llevaron a cabo a través de este programa de telemedicina, más de 42.000 consultas médicas. (Más información: <http://www.otn.ca/en/otn/who-we-are/>).

▪ **The Sophia Home**, una clínica privada en el centro de Estocolmo (Suecia) ha invertido en nueva tecnología ultramoderna que aumenta la eficacia y reduce el riesgo de infección. Las nuevas cámaras y pantallas mejoran y simplifican las operaciones y el uso de un nuevo sistema de videoconferencia permite a otros médicos y estudiantes atender a una operación sin necesidad de estar presentes. Al usar el nuevo sistema de sonido y vídeo, se pueden enlazar varias salas de conferencia para seguir el progreso de la sala de operaciones. El cirujano puede pedir consejo a otro médico que puede seguir las imágenes de la operación mediante una cámara endoscópica y dos cámaras montadas en el techo que pueden controlarse a distancia. Este otro médico puede proporcionar información mediante un PC conectado al quirófano, de otro modo dicho médico habría tenido que ir al quirófano y cambiarse de ropa para mantener una conversación de quizás 30 segundos. (Más información: <http://www.midlandcommsvc.co.uk/>).

▪ **El Ministerio de Defensa español** es uno de los organismos punteros en el desarrollo y aplicación en casos reales de la videoconferencia en la telemedicina. En la Sanidad Militar Española, la Telemedicina se desarrolla fundamentalmente en dos campos:

- La interconexión por Banda Ancha de todos los Hospitales de la Red Sanitaria Militar y algunas bases militares.
- La conexión con “Unidades de Sanidad” de los Ejércitos/Armada proyectadas internacionalmente.

Se denominan “Centros Remotos” a los equipos de Telemedicina que se encuentran ubicados fuera de la Red Hospitalaria de Defensa como actualmente son: los buques de la Armada con capacidad hospitalaria, las Unidades Sanitarias denominadas de tercer escalón de los Ejércitos/Armada como los EMAT (Escalones Médicos Avanzados del Ejército de Tierra), las UMAAD (Unidades Médicas de Apoyo Aéreo al Despliegue del Ejército del Aire) o las USANEM (Unidades de Sanidad Embarcadas de la Armada), que se encuentren proyectadas en misiones y que requieran para desarrollar la telemedicina conexiones vía satélite con la Red Hospitalaria de Defensa.

Estas conexiones se realizan por medio del sistema de comunicación vía satélite INMARSAT. Las terminales tipo M4 permiten una conexión de 64 Kbps de ancho de banda cada uno, lo que aporta un ancho de banda de 128 Kbps con 2 terminales satélite (los utilizados en la actualidad por las Unidades de Sanidad proyectadas).

Este ancho de banda vía satélite se utiliza para videoconferencia/imágenes: Ecográficas/Exploraciones/ Dermatoscopia/Otorrinoscopia en tiempo real, o 64 Kbps de videoconferencia junto a 64 Kbps de transmisión digital de datos sanitarios como radiografías, imágenes digitalizadas en color o monitorización de signos vitales, o Electrocardiografía de 12 derivaciones en tiempo real. Un desarrollo logrado en las últimas fechas, consistió en dotar a los buques de la Armada con terminales tipo F77 que disponen de una capacidad de ancho de Banda de 128 Kbps, con las mismas capacidades que los equipos de las Unidades anteriormente descritas. Todo ello permite el realizar tele-consultas urgentes o programadas con los Hospitales de la Red de Defensa, de modo que:

a) Se asesore a Oficiales Médicos Generalistas u Oficiales Enfermeros destinados en estas Unidades de los Ejércitos Armada, por parte de los Oficiales Médicos Especialistas de la Red Hospitalaria.

b) Se proporcione una asistencia con un 'plus' de calidad al personal desplegado (objetivo de la Sanidad Militar), debido a poder disponer de segundas opiniones en tiempo real.

(Más información:

[http://www.csi.map.es/csi/tecniap/tecniap\\_2006/05T\\_PDF/sistema%20de%20telemedicina.pdf](http://www.csi.map.es/csi/tecniap/tecniap_2006/05T_PDF/sistema%20de%20telemedicina.pdf))

▪ También en la **sanidad civil española** la tecnología se ha instalado, para quedarse. En los quirófanos del **Hospital de Puerto Real** se ha instalado en Abril de 2009 un novedoso sistema de videoconferencia quirúrgica por el que un conjunto de cámaras emitirá, en tiempo real y en alta definición, las operaciones quirúrgicas que se realicen en el centro. Este sistema, el primero de su estilo que se instala en la provincia, tendrá como función principal servir a los alumnos de medicina que realicen sus prácticas en el centro para contemplar cómo se realiza una operación, pero también podrá emplearse para transmitir dichas imágenes a otro hospital para que, desde éste, se pueda asesorar a los médicos en su intervención.

La principal novedad de este sistema es que permite la comunicación de los alumnos que estén viendo la intervención con el equipo médico que lo esté realizando en tiempo real. Del mismo modo, el grupo de cámaras que lo componen podrá controlarse exteriormente.

(Más información:

[http://www.lavozdigital.es/cadiz/20090415/puerto\\_real/camaras-toman-quiropano-20090415.html](http://www.lavozdigital.es/cadiz/20090415/puerto_real/camaras-toman-quiropano-20090415.html))

▪ **Hospitales de Alicante y Vega Baja** (Orihuela) realizan sesiones clínicas de patología cerebrovascular por videoconferencia. Para ello se utiliza el sistema de videoconferencia presente en todos los hospitales de la red de la Agencia Valenciana de la Salud, al que se ha añadido un software especial. De esta forma, los facultativos especialistas pueden estar interconectados a través de la web y su software, que permite la mutua visualización de los asistentes a la sesión clínica, la creación de un escritorio o área de trabajo común, compartir imágenes y su discusión.

La videoconferencia hace posible así el acercamiento de las sesiones clínicas multidisciplinarias entre servicios de hospitales alejados dentro del área sanitaria de Alicante, evitando el desplazamiento de los facultativos y facilitando la inmediatez en la discusión de los casos clínicos desde los respectivos servicios médicos. La primera sesión de teleconferencia inter-hospitalaria tuvo lugar el 22 de enero de 2008, durante

la cual se discutieron los casos de tres pacientes remitidos desde el Hospital Vega Baja al General de Alicante para la realización de un estudio por ictus cerebral. Desde entonces, los integrantes de la Comisión de Patología Cerebrovascular en ambos hospitales se reúnen semanalmente a distancia gracias a esta tecnología. (Más información: <http://www.neurorradiologia.org/sesiones/>)

### 2.3.2. PRECEDENTES EN EDUCACIÓN

El uso de videoconferencia para impartir educación y capacitación corporativa directamente en el lugar de trabajo ha sido la aplicación más exitosa y de mayor crecimiento de la videoconferencia. Tan sólo en España podemos encontrar miles de proyectos en los que la videoconferencia se utiliza como medio para educar. La combinación de esta tecnología con otros sistemas multimedia hace posible actualmente una oferta amplia de formación en las universidades y en otros centros de enseñanza, tanto en formación inicial como continua, de forma virtual.

La videoconferencia es un medio que facilita la comunicación bidireccional y simétrica ya que ambos extremos se convierten en emisores o receptores potenciales. Ya en 1995 el profesor Oliver M. de la Universidad de Baleares opinaba que "... tiene unas posibilidades educativas enormes, puesto que permite una interacción permanente, en tiempo real, [...] que no requiere grandes conocimientos técnicos para su manipulación, ya que su manejo es simple, transparente y porque su coste empieza a ser asequible."

La videoconferencia, entre las Tecnologías de la Información y de la Comunicación, ha sido la menos utilizada en el entorno educativo hasta un pasado reciente, aunque sí se emplea cada vez con más frecuencia, en el ámbito empresarial para los negocios y la administración. En el momento actual se está haciendo cada vez más popular en el área de la educación. El rápido desarrollo de sistemas informáticos con sistemas que permiten el uso de la videoconferencia (software específico y una cámara) a un precio relativamente bajo, ha introducido la video-enseñanza también en la clase, como una de las herramientas más prometedoras de la instrucción tecnológica en nuestros días.

A continuación mencionamos algunos ejemplos de *e-learning* en los se usa habitualmente la videoconferencia.

- **NASA: Proyecto Glenn de Tecnologías para el Aprendizaje** (1996-2000). Profesores y estudiantes de los niveles básicos tenían la oportunidad de hablar con científicos, investigadores, ingenieros y especialistas en educación de la NASA. Existía un calendario anual de sesiones de videoconferencia en el que se podían registrar los profesores. Cada evento tiene una página Web con una descripción del material que se sugería estudiar antes de la sesión, así como una lista de actividades que debían completarse. (Más información: <http://www.nasa.gov/centers/glenn/home/index.html>)

- **Acceso a Servicios de Interpretación Remotos.** La profesora Nanci Scheetz, que colabora en el programa de Interpretación del Lenguaje Americano de Signos, usa la videoconferencia para comunicarse con estudiantes con discapacidad auditiva

inscritos en la Universidad Estatal de Valdosta (Georgia, EEUU). La oficina proporciona un intérprete y otros personales de apoyo para tomar notas para el estudiante mientras la profesora se encuentra dando clase. Ambos servicios de interpretación y redacción de notas son críticos para los estudiantes con este tipo de discapacidad que deben enfocarse en el intérprete en señales, limitando el que hagan preguntas, participen en la clase y tomen notas al mismo tiempo. Para que los servicios de interpretación funcionen remotamente, tanto el estudiante como su intérprete poseen computadoras portátiles con equipo de videoconferencia conectados a una unidad multipunto. El mismo software que tiene la computadora del profesor es el que usa el estudiante sordo. A través de este software la Dra. Scheetz puede ver al instructor de la clase y al estudiante sordo con apoyo de la cámara conectada a la computadora portátil mientras el estudiante sordo ve en el monitor al intérprete, proporcionando comunicación por señales.

(Más información:

<http://www.valdosta.edu/coe/sec/ASLInterpretingandDeafEducation.shtml>)

- La UE está impulsando decididamente el e-learning a través del **Proyecto Knowledge On Demand (KOD)**, que se enmarca dentro del V Programa Marco de Desarrollo Tecnológico e Investigación de la Comisión Europea, que cuenta con una dotación de 2.300.000 euros y que está siendo desarrollado desde abril de 2000 por un consorcio compuesto por empresas públicas y privadas de cuatro países europeos, entre los que se encuentra España. El objetivo fundamental del proyecto KOD es desarrollar un portal vertical que incluya métodos personalizados de aprendizaje para los usuarios, incorporando herramientas de e-learning y de gestión del conocimiento.

En concreto, los impulsores son España, a través de la consultora ‘PROFit Gestión Informática’ y el ‘Centro de Alta Tecnología en Análisis de Imagen de Tenerife’ (CATAI), Grecia, a través del ‘*Informatics and Telematics Institute*’ (ITI) que es la institución que lidera el proyecto, Inglaterra, a través de ‘*FD Learning*’ y, finalmente, Italia con la compañía ‘ILABS’ (*Giunti Interactive Labs*).

(Más información: <http://kod.iti.gr/>)

- En el curso 2001-2002, la Universidad Autónoma de Barcelona le quitó el monopolio de la enseñanza virtual a la UOC (*Universitat Oberta de Catalunya*) con una licenciatura propia en Geografía completamente *on-line*. Hasta ese momento, esta universidad se había limitado a impartir algunos cursos de tercer ciclo para Latinoamérica, hasta que logró un acuerdo con las Universidades Carlos III de Madrid y Universidad de Alicante para crear, junto con la empresa del grupo Santillana-Santillana Formación, el **Instituto Universitario de Posgrado (IUP)**, con la intención de impartir másteres *on-line*. Estos programas están enfocados principalmente para licenciados en España y Latinoamérica que, al final del curso, reciben certificaciones de las tres universidades. Inicialmente esta agrupación impartía cuatro másteres: MBA, Finanzas, Nuevas Tecnologías Aplicadas a la Empresa y Nuevas Tecnologías Aplicadas a la Educación. (Más información: <http://www.iup.es>)

- En el año 2000, surge el Proyecto Intercampus que reúne a las universidades públicas catalanas, que comparten asignaturas de libre configuración. La **Universidad de Barcelona (UB)** que ofrece gran cantidad de Estudios de postgrado a distancia en los

que colabora con la UOC. La **Universidad Politécnica de Cataluña** (UPC) también posee un título propio de grado medio en el que colabora la UOC.

(Más información: <http://www.ub.es/acad/lliure/assignatures/intercampus.htm>)

▪ La **Universidad del país Vasco** puso en marcha una experiencia piloto de teleformación síncrona en el curso 1998/1999 empleando sistemas de videoconferencia. Para ello equipó cuatro salas ubicadas en Bilbao, Leioa, San Sebastián y Vitoria. Durante el primer semestre de 1999 las aulas fueron utilizadas para un total de 46 ponencias o lecciones, con una asistencia total estimada de 600 personas. Entre las aulas de videoconferencia existía conectividad ATM. (Más información: <http://www.rediris.es/difusion/publicaciones/boletin/50-51/ponencia14.html>)

▪ **Proyecto "Diseño Español" (DESPA):** El curso integrado de lenguas "Diseño español" (DESPA), se aprobó como parte de las actividades complementarias "ILCs": *Integrated Language Courses* en la primera edición del programa Sócrates (1997-98), recibiendo financiación durante tres años consecutivos, máximo periodo de renovación que este programa permite. El objetivo del curso era investigar las posibilidades del uso de las Tecnologías de la Información y de la Comunicación, especialmente de la videoconferencia, para el aprendizaje de segundas lenguas en las áreas específicas de diseño de moda, textil y gráfico. La presentación del proyecto docente ha consistido en un ciclo de 4 videoconferencias multi-punto, entre las 4 universidades componentes del proyecto y en las que se ha impartido un módulo de "Español para entornos profesionales: Presentaciones orales técnicas", módulo que incluía teoría y práctica, de tal manera que los estudiantes de diseño de las 4 universidades miembros, han presentado en español las obras reales de sus trabajos fin de carrera. En abril de 1997 se organizó la sesión piloto entre las dos universidades coordinadoras (Helsinki y Madrid), participando los coordinadores y profesores de ambas, y se convino el diseño, los objetivos los contenidos y actividades del proyecto así como la puesta en marcha del mismo. (Más información:

[http://cvc.cervantes.es/ensenanza/biblioteca\\_ele/ciefe/pdf/01/cvc\\_ciefe\\_01\\_0013.pdf](http://cvc.cervantes.es/ensenanza/biblioteca_ele/ciefe/pdf/01/cvc_ciefe_01_0013.pdf))

▪ La **Universidad de Málaga**, se inició en el *e-learning* llevando a cabo una experiencia piloto para los cursos de verano de 2002, que tuvieron una duración de casi cuatro meses, en las que sólo había que asistir a clases presenciales al primer día de presentación, siendo el resto del curso impartido exclusivamente por la Red. Estos cursos recibieron un claro impulso al dar la posibilidad, a aquellos alumnos sin conexión a Internet, de poder usar las aulas telemáticas de la UNED de Málaga.

(Más información: <http://campusvirtual.cv.uma.es/>)

▪ Un proyecto muy interesante que se está desarrollando en España es el proyecto ADA-Madrid. El **Proyecto ADA-Madrid** (Aula a Distancia Abierta) se puso en marcha en 2001 y está financiado por la Consejería de educación de la Comunidad de Madrid.

En el primer año del proyecto participaron las Universidades Autónoma de Madrid (UAM), Politécnica de Madrid (UPM), Universidad Carlos III de Madrid (UC3M) y Universidad Rey Juan Carlos (URJC), a las que se unieron posteriormente la

Universidad Complutense de Madrid (UCM) y la Universidad de Alcalá de Henares (UAH).

Su objetivo es la utilización de las TIC aplicadas a la educación para compartir asignaturas entre las universidades. El fin de este proyecto es que los alumnos puedan cursar asignaturas de libre configuración no disponibles en su universidad, pero sí en otras. La impartición se realiza vía Internet, con la ayuda de la videoconferencia y la distribución de vídeo a través de la red. Más de dos tercios de las asignaturas incorporan también clases telepresenciales que se emiten por videoconferencia IP (sobre Internet) a través de la red inter-universitaria de aulas de videoconferencia. De las cincuenta asignaturas impartidas en el curso 2007/2008, cuarenta y cuatro incorporaron alguna clase por videoconferencia, es decir, el 88%. Por último en lo que a los aspectos técnicos se refiere, durante el citado curso se ha implantado completamente el sistema de videoconferencia ISABEL, que durante el curso anterior se había empezado a utilizar en varias asignaturas. El sistema de videoconferencia IP, ha quedado como respaldo en caso de problemas técnicos. Atendiendo a este último aspecto conviene recordar que el modelo educativo de ADA-Madrid recomienda la utilización de la videoconferencia para facilitar la interacción de forma síncrona entre estudiantes y profesores y entre los propios estudiantes, reduciendo su utilización para clases expositivas.

(Más información: <http://moodle.upm.es/adamadrid/>)

▪ Otro proyecto es el **Campus Virtual del G-9**. El Grupo 9 de universidades es una asociación sin ánimo de lucro que agrupa a las universidades de las Islas Baleares, Cantabria, Castilla-La Mancha, Extremadura, La Rioja, País Vasco, Pública de Navarra, Oviedo y Zaragoza. El grupo fue constituido en el convenio firmado el 16 de mayo de 1997. Las Universidades del G9, tal y como reflejan sus estatutos, tienen como objetivo social común promover la colaboración entre las instituciones universitarias pertenecientes al Grupo, tanto en lo que respecta a las actividades docentes e investigadoras como a las de gestión y servicios. En 1999, el Grupo 9 de Universidades tomó la decisión de crear un “Campus Virtual Compartido”, con asignaturas que pudieran ser cursadas por cualquiera de los estudiantes del resto de universidades del grupo. El objetivo principal de este proyecto es potenciar la movilidad virtual de alumnos entre las universidades que forman dicho grupo y la creación de una oferta educativa conjunta basada en las nuevas tecnologías. Para lograr este objetivo se pensó en compartir asignaturas de libre configuración. En el curso 2003/2004 se impartieron 42 asignaturas.

Este proyecto se apoya en el uso de la videoconferencia y de la información difundida a través de Internet, y pretende:

1. Compartir asignaturas presentes en los planes de estudios de dos o más universidades del grupo, de manera que partes comunes de las asignaturas puedan ser ofrecidas a los estudiantes de algunas universidades de dicho grupo.
2. Facilitar el intercambio y la creación de equipos docentes dentro del grupo 9 de universidades, superando el marco local de cada universidad y fomentando una visión global y amplia de la docencia y el aprendizaje.
3. Mejorar el uso de los recursos docentes del G-9, optimizándolos en su caso, mediante el trabajo colaborativo y compartido de aquellos equipos que finalmente se seleccionen para llevar a cabo la experiencia.

4. Fomentar el uso de las TICs como herramienta de comunicación y como medio para acercar profesorado y estudiantes.

(Más información: <https://www.uni-g9.net/portal/faq.html>)

- Existe además el proyecto **Intercampus entre España y América Latina** que nace para aunar las universidades de ambos lugares. El Programa Intercampus de Cooperación Científica e Investigación Interuniversitaria entre España e Iberoamérica (PCI Intercampus), funciona como fuente de comunicación e intercambio que se suma a las relaciones que mantienen, desde hace años, las universidades españolas y latinoamericanas. El proyecto tiene como objetivo el fomento de vínculos estables de investigación y docencia entre España y los países de Ibero América, a través de actividades de cooperación científica e investigación para la realización de proyectos conjuntos de investigación, proyectos conjuntos de docencia y acciones complementarias.

Los primeros pasos de este proyecto se dieron en 1994, que propinaron la participación de 671 estudiantes de pregrado o postgrado españoles entre julio y septiembre de 1995, desde entonces las universidades latinoamericanas han recibido a miles de estudiantes y cientos de profesores. (Más información: <http://www.becasmae.es/pci/>)

- La **Universidad Carlos III de Madrid** durante el año 1998/1999 dentro del proyecto PROMETEO (financiado por la CICYT) se ha dotó de la infraestructura necesaria para la realización de tele educación y videoconferencia.

Esta dotación comprende: equipamiento de red necesario (conmutadores ATM, conmutadores Ethernet,..) adquisición de equipos CODEC para las distintas tecnologías a utilizar por parte de la comunidad universitaria (video sobre ATM, H.320, H.323 y Mbone), adquisición de equipamiento de audiovisuales (matrices de video, micrófonos, megafonía, equipos de control, cámaras,...) y creación y/o adecuación de salas de audiovisuales. Tiene en sus tres campus salas de audiovisuales especialmente preparadas para que en ellas se desarrollen videoconferencias. Las aulas virtuales de los tres campus están interconectadas a través de la red IP de la universidad. Además existen salas de Teledocencia, que son salas preparadas para la impartición de tele-educación y equipadas con sistemas de codificación y decodificación que permiten la transmisión y recepción de eventos en alta calidad (MPEG-2) y Windows Media. Desde estas salas se facilita la celebración de sesiones de vídeo y teleconferencia y la grabación en formatos digitales de clases, seminarios, charlas, etc. Además, como hemos visto, participa en la iniciativa ADA-Madrid y forma parte del Instituto Universitario de Posgrado. (Más información: <http://audiovisuales.uc3m.es/>).

- **Virtual Rooms Videoconferencing System:** VRVS significa “Sistema de videoconferencia basado en Salas Virtuales”. VRVS es una plataforma de colaboración entre personas geográficamente dispersas que funciona a través del sitio web: <http://www.vrvs.org>

VRVS es un sistema basado principalmente en videoconferencias multipunto (conexión de dos ó más personas al mismo tiempo). Funciona bajo redes IP (Internet Protocol) y soporta la mayoría de los sistemas operativos conocidos. VRVS es propiedad de Caltech



(*California Institute of Technology*) y su uso está orientado únicamente a las comunidades educativas y de investigación en el mundo.

La utilidad principal de este sistema es la comunicación entre estudiantes, profesores y/o investigadores que se encuentren separados geográficamente y necesiten colaborar entre ellos en cualquier momento y desde cualquier lugar.

El sistema VRVS se compone de dos partes bien diferenciadas:

- el servidor web en donde los usuarios se conectan a las videoconferencias y lanzan sus aplicaciones
- una red mundial de reflectores interconectados que distribuyen los flujos de información a cualquier lugar desde el que el usuario se encuentre conectado. A fecha de febrero de 2003 la red estaba formada por 61 reflectores ubicados en 22 países distintos.

Cuando un usuario se conecta al sistema VRVS, su máquina queda asociada automáticamente al reflector más próximo o al que tenga una mejor conexión. Siempre que este usuario envíe audio, vídeo o datos, lo hará su reflector asociado y cuando reciba audio, vídeo o datos, lo recibirá igualmente de su reflector asociado.

### **2.3.3. PRECEDENTES SIMILARES**

Los ejemplos que se han citado en el apartado 2.3.2, podemos observar que tienen una característica en común: ambos extremos de la comunicación están situados en una habitación adecuadamente preparada para el evento. En la mayoría de los mencionados ejemplos se han usado materiales y tecnologías tales como: *codecs* para H.320 y H.323 ó *codecs* para transmitir vídeo de alta calidad sobre ATM, conmutadores ATM y Ethernet, el caso de las multiconferencias se ha necesitado una MCU, micrófonos de ambiente, aplicaciones Mbone para multicast... Además en estos lugares también se disponía de las instalaciones de red pertinentes, teniendo, por tanto, fácil acceso a la red por la que se iba a transmitir la información: normalmente ISDN o ATM. Obteniendo de esta forma videoconferencias de alta calidad.

La situación que en este proyecto se plantea es algo diferente. Es cierto que uno de los extremos de nuestra comunicación es, al igual que en los ejemplos anteriores, una habitación, que podrá estar equipada con las tecnologías que creamos convenientes. Pero el otro extremo debe resolverse de forma diferente, pues será una sala no habilitada para videoconferencias y sin ningún tipo de material relacionado con éstas. Puede ser, por ejemplo, un museo de pintura o puede ser simplemente un espacio al aire libre en medio de un paraje natural. La clave está en que el extremo remoto pueda ser lo que queramos. Es decir, el proyecto trata de buscar una solución de videoconferencia en la que uno de los extremos pueda estar situado en cualquier lugar o entorno que se nos ocurra.

Es por ello que hemos investigado más a fondo para encontrar casos que pudiesen tener mayor parecido con el proyecto que hemos planteado.

### **2.3.3.1. Precedente similar en Educación**

Tal y como pudimos leer en el periódico “Extremadura al Día” del 12 de Diciembre de 2006, el Instituto de Educación Secundaria ‘Hernández Pacheco’ llevó a cabo una original experiencia con sus alumnos, estableciendo una conexión por videoconferencia con el Hospital San Pedro de Alcántara. Conocer este medio de comunicación era uno de los objetivos. Los alumnos de este instituto de Cáceres disfrutaron de esta actividad el día 13 de Diciembre de 2006. La actividad se enmarcó en el programa de actos organizados para conmemorar el cincuentenario de la puesta en marcha del centro hospitalario.

Otro de los objetivos de esta iniciativa era que los jóvenes estudiantes conociesen el mundo hospitalario a través de sus profesionales, en directo y en red.

Los alumnos asistentes tuvieron la oportunidad de formular sus preguntas, que fueron respondidas tanto por el propio gerente, como del resto del personal sanitario, quienes aclararon las dudas sobre el funcionamiento del Hospital San Pedro de Alcántara de Cáceres, desde la hospitalización de pacientes, tramitación e incluso las posibilidades y requisitos para entrar a trabajar en el mismo. (Más información: <http://www.extremaduraaldia.com/tecnologia/el-ies-hernandez-pacheco-de-caceres-se-conectaran-por-videoconferencia-con-el-hospital-san-pedro-de-alcantara/29629.html>).

Con esta experiencia se pretendía lograr que los alumnos se acercasen tanto a las nuevas tecnologías como a una realidad fuera del instituto. Este objetivo es el mismo que se pretende con nuestro proyecto, que los alumnos puedan hacer algo así como ‘excursiones virtuales’ que les permita observar el lugar de interés en primera fila sin interferir, por ejemplo como en este caso, en el funcionamiento del hospital, pero obteniendo la misma información audiovisual que si estuviesen allí presentes. Las ventajas que este sistema presenta para los profesores son enormes, pues evita tener que desplazarse con treinta o cuarenta alumnos al lugar de interés con el esfuerzo que ello conlleva, sobretodo si los alumnos son de corta edad. En la mayoría de casos, además, este tipo de visitas obligaría al colegio o instituto a contratar un servicio de transporte para los alumnos hasta el lugar de destino.

Pero esta metodología también presenta ventajas para los alumnos. Al facilitarse para los profesores la tarea de acercar otras realidades a la propia aula, probablemente más profesores se animen a realizar habitualmente ‘excursiones virtuales’. Los alumnos podrán, por tanto, disfrutar más frecuentemente de actividades relacionadas con lugares fuera del centro de estudios.

Sin embargo, en la experiencia llevada a cabo en el Instituto Hernández Pacheco la infraestructura tecnológica empleada sigue siendo idéntica a los casos que hemos visto en el apartado 2.3.2. En ese momento el centro hospitalario cacereño llevaba ya algún tiempo realizando pruebas de telemedicina, usando la videoconferencia y disponiendo de costosas instalaciones dedicadas a tal tarea situadas en un lugar fijo dentro del hospital. Estas instalaciones del hospital fueron las que se utilizaron para llevar a cabo la citada videoconferencia. Por tanto, en esta experiencia que se realizó entre el hospital y el centro de estudios, ambos extremos de la comunicación disponían de instalaciones fijas para la comunicación a través de videoconferencias.

En el proyecto que estamos realizando, queremos que un extremo de la comunicación sea completamente móvil. Nuestro proyecto, por tanto, coincide con el proyecto del

Instituto Hernández Pacheco en la metodología educativa que se utiliza pero no en el objetivo puramente tecnológico que se plantea.

Si nos basásemos en los objetivos de nuestro proyecto y los aplicásemos al caso que estamos mencionando, la idea sería que los alumnos del instituto hubiesen podido realizar la “tele-visita” al Hospital San Pedro de Alcántara aunque éste no hubiese estado dotado de infraestructura alguna de telecomunicaciones. De este modo se habría podido organizar una visita igual al centro de salud del barrio o al parque de bomberos de la ciudad.

Es decir, la solución tecnológica que se le ha dado al proyecto de Cáceres da solución únicamente a la situación concreta en la que aparecen como actores ese instituto y ese hospital. Nuestro proyecto, sin embargo, trata desarrollar una solución mucho más genérica que daría solución ese mismo problema de Cáceres pero también deberá dar solución al problema fuese cual fuese el actor en remoto.

### **2.3.3.2. Precedentes similares en la Televisión**

En nuestro proyecto, tal y como sabemos, la situación que se plantea es una videoconferencia entre una habitación equipada para llevar a cabo una videoconferencia y un lugar arbitrario. En esta sociedad dominada por las tecnologías y los medios de comunicación no es difícil percatarse que el escenario que estamos planteando es el mismo que vemos todos los días en televisión. Los “Informativos” suelen conectar una o varias veces en cada edición con los periodistas que se encuentran en el lugar de la noticia, dondequiera que ésta esté teniendo lugar. Otros programas, asimismo, muy de moda en los últimos años, emplean toda su duración en conexiones en directo con los reporteros que tienen repartidos por toda una Comunidad (como por ejemplo “Madrid Directo”, en Telemadrid) o por todo un país (“España Directo”, en TVE1).

La transmisión en directo de la señal de televisión requiere actualmente de unidades móviles de enlaces o conexión por satélite que es necesario reservar de forma anticipada y de elevado coste económico. Esto dificulta enormemente la transmisión de informaciones fuera de plató y la transmisión en directo de noticias que suceden de forma imprevista. A ello se une además las dificultades para acceder a determinados lugares, como interiores de edificios o espacios con grandes aglomeraciones, ya que las cámaras de televisión están unidas por cable a la unidad móvil.

En los apartados anteriores hemos revisado varias experiencias concretas de uso de videoconferencia en educación. Tan sólo uno de los casos revisados presentaba mayor similitud con este proyecto (apartado 2.3.3.1) y se trataba una similitud a nivel de la situación educativa experimentada pero no daba solución a nuestro problema. Es en un entorno diferente, el de los medios de comunicación y, concretamente, en la televisión donde encontramos experiencias que se aproximan en mayor modo al esquema técnico que nosotros queremos implementar.

En las experiencias que se van a mencionar a continuación se ha usado la videoconferencia como medio para transmitir noticias en un contexto de un programa informativo. Nos puede parecer a priori que este entorno y el de nuestro proyecto tienen poco que ver, pero no es así. Educar es transmitir conocimientos y los conocimientos se transmiten a través de la información. Una noticia es información antes desconocida. Esta relación directa entre los conceptos ‘educación’, ‘información’ y ‘noticia’ nos hace

intuir que no sólo van a aparecer similitudes de tipo técnico, también similitudes en cuanto a la esencia de la información que se transmite en ambos casos.

### ▪ **Primera prueba en España: Barcelona Televisió**

En Julio de 2007 Barcelona Televisió y Telefónica I+D transmitieron una señal de televisión en directo sin necesidad de unidades móviles. La tecnología desarrollada por el Centro de Barcelona de Telefónica I+D permitió sustituir una unidad de enlaces terrestres o satélite por un equipo que controlaba varios teléfonos móviles.

Tal y como se puede leer en “noticias.info” (y más concretamente en la página web: [http://www.noticias.info/archivo/2007/200707/20070727/20070727\\_303739.shtm](http://www.noticias.info/archivo/2007/200707/20070727/20070727_303739.shtm)), ambas empresas llevaron a cabo con éxito la primera experiencia de transmisión de televisión en directo, en pantalla completa, sin usar una unidad de enlaces de televisión. El programa “Hola Barcelona” emitió el día 26 de julio de 2007 en directo un reportaje de veinte minutos de duración, sin necesidad de desplazar, como es habitual, una unidad de enlaces terrestres o satélite para capturar y transmitir en directo la señal audiovisual.

La tecnología WENG (*Wireless ENG*) desarrollada por Telefónica, hacía posible capturar y transmitir en directo la señal audiovisual en pantalla completa desde cualquier lugar en el que existiese cobertura UMTS, la tecnología móvil de tercera generación. Esto se conseguía mediante un software inteligente que agrega el ancho de banda de varios teléfonos UMTS, y coordina el tráfico en función de la velocidad que se consigue en cada conexión móvil. El equipo básico para estas nuevas unidades móviles, cámara y micrófono aparte, consta de dos ordenadores portátiles y cuatro teléfonos móviles con acceso a Internet. El primer portátil comprimía la imagen que captaba la cámara, mientras que el segundo fragmentaba la señal y la enviaba por paquetes a través de los móviles. Cuando la señal llegaba al estudio central, el proceso se hacía a la inversa, uniendo los paquetes y descomprimiéndolos para su inmediata emisión. Hasta ese momento, algunas cadenas de televisión norteamericanas realizaban transmisiones en directo utilizando UMTS, pero en pantalla pequeña.

La tecnología WENG no utiliza cables, por lo que las cámaras de televisión podían moverse sin dificultades durante la transmisión de un directo, y sólo requiere de varias conexiones UMTS para transmitir la señal, con el consiguiente abaratamiento de los costes. El equipo WENG realizaba automáticamente llamadas usando la tecnología de mayor capacidad disponible en cada zona (HSUPA, HSDPA o UMTS) para permitir en cada momento a la unidad móvil transmitir imagen y sonido con la máxima calidad posible.

El objetivo de Telefónica era que esta tecnología, además de su aplicación en el campo audiovisual, pudiese asimismo exportarse a otros como pueden ser la seguridad ciudadana, la mejora de la cobertura en situaciones de emergencia o en aplicaciones móviles para el usuario final. Lo cierto es que casi dos años después de que se realizaran estas pruebas no se ha vuelto a saber nada más sobre esta tecnología.

### ▪ **Prototipo Universidad de Sevilla y Canal Sur**

El Grupo de Investigación de Tratamiento de Señales y Comunicaciones de la Escuela Técnica Superior de Ingenieros de la Universidad de Sevilla (que dirige el profesor J.

Ramón Cerquides Bueno), en colaboración con Canal Sur Televisión, presentaron en Febrero de 2009, una tecnología mediante la cual es posible realizar grabaciones de televisión en directo o en falso directo sin necesidad de contar con una unidad móvil convencional, como es habitual en la actualidad.

Tal y como podemos leer en la página web de “Universia” ([http://www.universia.es/html\\_estatico/portada/actualidad/noticia\\_actualidad/param/noticia/jifdb.html](http://www.universia.es/html_estatico/portada/actualidad/noticia_actualidad/param/noticia/jifdb.html)), el prototipo aprovecha el alto nivel de compresión que alcanzan actualmente los formatos digitales de vídeo, algo que, unido a la velocidad de transmisión de datos que proporcionan las redes UMTS, permite que la unidad móvil, cuya labor hasta ahora es indispensable para el transporte de la señal hasta los estudios centrales de televisión, sea sustituida por un módem 3G (de tecnología móvil) de transmisión de datos.

De esta forma, únicamente habría que conectar el equipo de grabación a un ordenador portátil con el que se comprimirían y codificarían las imágenes en bruto, tras lo cual los datos resultantes serían remitidos a través de un módem HSUPA hasta la estación base de telefonía móvil, y de ahí a los estudios centrales, vía Internet.

Hoy, la grabación de imágenes de vídeo en exteriores se deja en manos de los equipos de Producción Electrónica de Informativos (ENG, *Electronic News Gathering*), dedicado a la producción de noticiarios, reportajes o entrevistas, y los de Producción Electrónica en Exteriores, que normalmente llevan a cabo documentales. La labor de un equipo ENG normalmente consiste en tomar imágenes en bruto que luego son editadas en los estudios centrales mediante diversos programas informáticos, con los que se realiza una selección de las mejores imágenes, y se les añade voz y otros efectos. Esto sucede si se trata de una información que será retransmitida con posterioridad, mientras que si se trata de información en directo, el periodista y el cámara han de desplazarse hasta el lugar del hecho junto a una unidad móvil que permite la transmisión de las imágenes hasta los estudios mediante un sistema de microondas o vía satélite. Este procedimiento es lento y complicado, aunque en menor medida si se trata de una información en directo pero de corta duración, caso en el que se usan Unidades Móviles Ligeras. Éstas unidades, de menor tamaño, son las susceptibles de ser sustituidas por el nuevo sistema de transmisión ideado por los investigadores.

#### ▪ Programa en antena: “Los corresponsales en 24 horas”

Por el momento, el único programa en emisión en España que utiliza como medio único para la conexión con sus corresponsales la videoconferencia es “Los Corresponsales en 24 horas”. Dicho programa se emite, desde Marzo de 2009, todas las tardes de lunes a jueves por el ‘Canal 24 Horas’ de TVE. Éste programa apuesta por un formato informativo innovador. Cuenta con la colaboración de los 16 corresponsales de TVE y en cada programa algunos de ellos establecerán una red de comunicación en directo a través de videoconferencia.

Además, el programa tiene un blog (<http://blogs.rtve.es/corresponsales/posts>) desde el cual la propia presentadora, la periodista Almudena Ariza, adelanta cada día los temas que se van a tratar esa misma tarde, pudiendo los lectores del blog participar con preguntas escritas o vídeopreguntas, que pueden enviar al programa. Asimismo se pueden proponer temas o mantener conversaciones escritas con Almudena Ariza o con los corresponsales a través del propio blog.

“Los Corresponsales en 24H” se presenta en su propio blog del siguiente modo:

*“Internet llevado a la televisión para conectarnos al mundo. Eso es Corresponsales en 24 Horas. Acercar realidades lejanas con la mirada siempre cercana de los 16 periodistas que TVE tiene desplazados por todo el mundo.*

*Están en Bogotá, Buenos Aires, La Habana, México, Washington, Nueva York, Pekín, Moscú, Berlín, Bruselas, París, Rabat, Jerusalén, Londres, Roma y Lisboa. Cada tarde de lunes a jueves, a las 18.30 (hora española) Almudena Ariza conectará con los corresponsales por videoconferencia. [...]. TVE es pionera en España en el uso de Internet para informar en directo. Ellos nos contarán qué es noticia en sus países y nos lo explicarán de la forma en que sólo quien vive allí puede hacerlo.”*

Cada tarde, los corresponsales elegidos, normalmente cinco de ellos, mantienen una videoconferencia con el plató de TVE vía Internet, pero aunque todos se comunican a través de videoconferencia, los medios que utiliza cada corresponsal para llevar a cabo dicha conexión varían.

En ocasiones el corresponsal aparece en una habitación en la que parece tener todas las facilidades para conectar con Madrid. Aparece en un escritorio, sentado y por la calidad de la imagen mostrada se podría deducir que disfruta de una buena conexión a Internet y de una cámara de calidad.

Por el contrario en otras ocasiones, por el contexto de la noticia, el corresponsal debe aparecer en un exterior, de pie y con un ordenador portátil, que va girando sobre sus manos en función de la imagen que quiera mostrarnos a través de la propia webcam que suele ir integrada en los ordenadores portátiles. En estos casos la conexión a Internet han de realizarla a través de un modo de conexión inalámbrico, probablemente tecnología WiFi o 3G. En este caso la calidad de la imagen recibida disminuye notablemente con respecto a la calidad obtenida en el entorno fijo mencionado en el párrafo anterior.

“Los Corresponsales en 24 horas” sigue un formato innovador en España, al que probablemente le saldrán muchos imitadores. Quizá no inmediatamente, pero en pocos años, cuando las tecnologías necesarias hayan mejorado, es probable que la mayoría de programas que están basados en un formato en el que la conexión con corresponsales es habitual, migren hacia este modelo de conexión a través de videoconferencia

### **2.3.3.3.- Reflexión acerca de los precedentes similares en la televisión**

Las ventajas que presenta un sistema que sustituya las unidades móviles de televisión son innegables: supone un enorme ahorro de costes al no ser necesario el uso de un Unidad Móvil de Televisión que requiere una inversión inicial de una cuantía importante y un coste adicional cada vez que se ha de desplazar la Unidad para cubrir una noticia, además requiere de mínimo 5 personas dedicadas a ello.

Con formatos televisivos que basen sus emisiones en directo en la videoconferencia, como ocurre en el programa “Los Corresponsales en 24 Horas”, una sola persona puede llevar a cabo la transmisión con un material mínimo compuesto de: ordenador portátil con cámara integrada, conexión a Internet y acceso a algún software de videoconferencia.

Para que de verdad se adopte en la mayoría de programas informativos el modelo que acabamos de estudiar en el que las unidades móviles se ven sustituidas por soluciones de videoconferencia, deberá avanzar la tecnología en dos aspectos fundamentales:

- Se deberá diseñar un dispositivo de pequeño tamaño, fácilmente transportable y capaz de conectarse a Internet a través de una red inalámbrica de banda ancha. El dispositivo deberá estar dotado de una cámara de vídeo que proporcione una imagen de gran calidad (comparable a una cámara de vídeo profesional) y un micrófono que proporcione un excelente sonido.
- Será necesario que haya facilidad para acceder sin cables a Internet en cualquier lugar del mundo, tanto en interiores como en exteriores, a gran velocidad. Hay que tener en cuenta que se deberá disfrutar de un ancho de banda lo suficientemente amplio como para transmitir sin problema ese audio y vídeo de gran calidad que hemos mencionado, en tiempo real.

Tal y como mencionábamos en el Capítulo 1, nuestro proyecto pretendía llevar a las aulas en cierto modo el estilo de un “riguroso directo televisivo” a través de la videoconferencia. En “Los corresponsales en 24 Horas” se ha conseguido llevar a la televisión el “riguroso directo televisivo” a través de la videoconferencia. Observamos que la situación es idéntica excepto por el extremo en local, que en nuestro proyecto es un aula y en el del canal 24 Horas es un estudio de televisión. En “Los corresponsales en 24 Horas” al parecer los corresponsales en remoto acceden a Internet de diferente modo dependiendo de las tecnologías disponibles en cada lugar. Sin embargo, nuestro proyecto trata de utilizar la misma tecnología sea cual sea el emplazamiento remoto.

En este aspecto, quizá se asemejen más a nuestro proyecto las experiencias de Barcelona Televisió y Canal Sur, pues en ambos casos se ha optado, al igual que en nuestro proyecto, por buscar una única solución para resolver la conexión sea cual sea la situación geográfica del extremo remoto. Sin embargo, en estos dos casos no se menciona que se haga uso de la videoconferencia, más bien se trataría de una solución inalámbrica para enviar vídeo desde el extremo remoto hasta los estudios de televisión de modo que estos pudiesen ser emitidos en directo.

## **2.4. PROGRAMAS DE VIDEOCONFERENCIA EXISTENTES.**

En este apartado vamos únicamente a mencionar algunos de los programas de videoconferencia sobre Internet disponibles en la actualidad. Nos centraremos únicamente en aquellos programas que son gratuitos. Mencionaremos muy brevemente sus características más relevantes y los inconvenientes principales que encontramos.

Clasificaremos estos programas en dos tipos: aquellos que requieren instalación y aquellos que no la requieren. A éstos últimos se les denomina “basados en Web”.

### **2.4.1. PROGRAMAS DE VIDEOCONFERENCIA EN LOS QUE ES NECESARIA INSTALACIÓN**

Estos son los programas de videoconferencia clásicos en los que hemos de tener un fichero ejecutable que lanzaremos en el ordenador que vayamos a utilizar para llevar a cabo la videoconferencia. Este ejecutable instalará, por tiempo indefinido en nuestra máquina, el software de videoconferencia. Algunos ejemplos son:

#### · Microsoft Net Meeting

Se trata de un cliente de videoconferencia VoIP multipunto incluido en muchas versiones de “Microsoft Windows” antiguas. Desde el lanzamiento de “Windows XP”, Microsoft lo abandonó en favor de “Windows MSN Messenger”. Implementaba una solución de videoconferencia sobre Internet con chat, pizarra electrónica, posibilidad de compartir el escritorio o transferir archivos, aparte de la transmisión punto a punto del audio y el vídeo. No existía la posibilidad de grabar la videoconferencia.

#### · MSN Messenger

MSN Messenger fue el relevo de “Microsoft Net Meeting” es un programa de mensajería instantánea creado en 1999 y actualmente sustituido por “Windows Live Messenger”. La primera versión de MSN Messenger que incluyó videollamada fue la 7.0, lanzada en abril de 2005. Dicha versión cuenta con chat, videollamada, conversación de voz, etc. Este programa fue renombrado como “Windows Live Messenger” a partir de la versión 8.0. Tampoco permitía la grabación de la videollamada.

#### · Windows Live Messenger

Windows Live Messenger nace el 13 de diciembre del 2005. Ofrece, como sus antecesores, servicios de videollamada de ordenador a ordenador, pero ahora incluso con imágenes a pantalla completa. El principal inconveniente continúa siendo la incapacidad de grabar las videollamadas.



### · Skype

Skype es un exitoso programa de videollamadas que proporciona a los usuarios una imagen de gran tamaño de su interlocutor y chat. Es muy sencillo y fácil de usar. Se puede asimismo compartir el escritorio y enviar archivos. Destaca la calidad del audio. No permite la grabación de las videollamadas.

### · Conference XP

Se define como una herramienta de trabajo “colaborativo”, desarrollada por Microsoft, para su uso en distintos entornos: *e-learning*, colaboración remota, videoconferencia, etc. Además de la pura videoconferencia permite compartir el escritorio, realizar presentaciones “Power Point” o enviar archivos “Windows Media”. Para disfrutar de otras funcionalidades es necesario instalar herramientas adicionales que se apoyan en este software a través de la cuales podremos, entre otras muchas opciones, almacenar y distribuir mediante *streaming* la sesión de videoconferencia. Hay que instalar el software correspondiente en el servidor, el cliente, el reflector y el servidor de archivos.

Si nos planteásemos utilizar una herramienta de videoconferencia de este tipo en nuestro proyecto, encontraríamos varios problemas: es un programa nada amigable para el usuario estándar, no tiene una interfaz gráfica intuitiva. Poner en marcha un sistema así requiere la instalación previa de software en varias máquinas y su configuración por parte de técnicos entendidos en la materia. El disfrute de herramientas como la grabación de la sesión requiere la instalación de software adicional. En definitiva, se trata de un programa que no ha sido pensado para el público en general sino más bien a comunidades con expertos en informática y telecomunicaciones. Es un programa orientado a las comunidades docentes e investigadoras.

Otros programas de videoconferencia muy similares a éste y que tan sólo vamos a mencionar son: VRVS(*Virtual Rooms Videoconferencing System*), Access Grid o Isabel.

### · ooVoo

Programa de videoconferencia de escritorio que requiere instalación. La versión gratuita permite realizar videoconferencia entre dos usuarios, enviar archivos de gran tamaño, grabar video-mensajes de 1 minuto de duración y dispone de chat.

Es muy intuitivo y fácil de usar, además el tamaño y la calidad del vídeo enviado/recibido es bastante buena. Sin embargo la versión gratuita no incluye la opción de grabar la videollamada. Una de las versiones de pago te permite grabar el vídeo e incluso colgarlo en la página web que se le indique.

### · VSee

Es un software colaborativo de videoconferencia que además permite compartir aplicaciones, el escritorio, archivos, dispositivos y ofrece chat. Es gratuito siempre y cuando se llame a menos de diez personas al mes. Una de sus mayores ventajas es que requiere muy poco ancho de banda. Además permite grabar la sesión y expandir lo que se quiera la imagen del locutor. La instalación es extremadamente sencilla.

· Otros: Qnext, SightSpeed, Yahoo! Messenger, Ekiga (Gnome Meeting), KPhone, Linphone, Mercurio IMS Client, Minisip.

### **2.4.2. PROGRAMAS DE VIDEOCONFERENCIA BASADOS EN WEB**

En inglés se suele conocer como “*web-based conferencing*” refiriéndonos a aquellos tipos de videoconferencias en los que no es necesario instalar previamente ningún software en el ordenador en el que ésta se va a llevar a cabo. Esta clase de programas están normalmente pensados para que cualquier persona que quiera realizar una videoconferencia, independientemente de sus conocimientos informáticos, sea capaz de realizarla con facilidad. El usuario tan sólo debe registrarse en la página web del producto y conservar su usuario y contraseña. Cada vez que este usuario quiera realizar una videoconferencia le bastará con acceder a esa página y meter esos datos. Suelen tener una interfaz gráfica muy sencilla y amigable para cualquier usuario. Algunos ejemplos son:

#### · Dimdim

Es un programa de videoconferencia basada en Web que en su versión más sencilla es gratuito. Se lanzó en Febrero de 2006. Es muy fácil de usar y permite, además de la videoconferencia en sí, compartir fotos, páginas Web, presentaciones “PowerPoint”, “PDFs”, tu propio escritorio, una pizarra, incluso vídeo durante la propia videoconferencia. Además tiene chat. Es muy flexible y muy fácil de usar. Asimismo se pueden grabar las sesiones y al finalizar la videoconferencia te envían un mail facilitándote el enlace permanente a esa grabación. La grabación contiene el vídeo con la imagen del hablante y los mensajes del chat.

Es un programa muy completo pero con un inconveniente y es que la imagen de los locutores es muy pequeña.

#### · Vyew

Es otro de los programas de videoconferencia basada en web. Además de la transmisión de audio y vídeo, dispone de chat, se puede compartir el escritorio, compartir documentos, usar una pizarra, reproducir contenidos multimedia... además pueden participar muchos usuarios a la vez.

Una de sus mayores ventajas es que todo el contenido de las sesiones queda grabado, y pueden acceder a él permanentemente los usuarios autorizados. Además se puede directamente publicar dicha grabación en una página web o enviarla por mail.

El mayor inconveniente que presenta, como el programa anterior, son las reducidas dimensiones en las que se presenta la imagen de los participantes.

#### · PalBee

Se trata de nuevo de una herramienta de videoconferencia basada en web, por lo que para comenzar a usarla tan sólo es necesario registrarse en su página web. Es muy parecido al anterior programa, pues la sesión puede ser grabada y se almacena en la web, sólo podrán acceder a ella los usuarios autorizados. Podemos compartir una pizarra, se pueden compartir presentaciones “Power Point” e imágenes y dispone de chat.

De nuevo el principal problema son las reducidas dimensiones en las que se muestran las imágenes capturadas por la webcam.

· FlashMeeting

Es un programa gratuito de videoconferencia para los miembros de la comunidad de desarrollo “Open Learn”. Cualquier usuario se puede registrar y participar en los cursos o realizar videoconferencias educativas. También se dispone de canal de chat público y privado, pizarra y la posibilidad de compartir páginas web e incluso grabar la videoconferencia. La interfaz, de cara al usuario, no es demasiado intuitiva y, de nuevo, la imagen del locutor tiene unas dimensiones demasiado reducidas.

· Otro: Tokbox.

## **CAPÍTULO III**

### **ESTADO DEL ARTE: WiMAX Worldwide Interoperability for Microwave Access**

La movilidad es un requisito básico que debe cumplir nuestro proyecto. Deseamos que uno de los extremos de la comunicación pueda estar situado prácticamente en cualquier lugar. Necesitamos, por tanto, hacer uso de algunas de las tan populares en la actualidad, tecnologías inalámbricas. La gran ventaja de este tipo de tecnologías reside en la movilidad y ubicuidad que proporcionan. Su éxito en los últimos años ha provocado no sólo un gran interés por parte del público, sino también por parte de las compañías, que se han apresurado a desarrollar multitud de tecnologías capaces de transmitir información sin necesidad de cables.

Una de las tecnologías más novedosas en este ámbito se denomina WiMAX (Worldwide Interoperability for Microwave Access) y es la tecnología elegida en este proyecto para dar solución a la transmisión de audio y vídeo entre un entorno móvil desconocido, al que llamaremos extremo remoto, y un entorno fijo bien definido, al que llamaremos extremo local.

A lo largo de este capítulo explicaremos exactamente en qué consiste WiMAX, las tecnologías y estándares que lo definen, sus características principales, así como lo compararemos con otras tecnologías inalámbricas de banda ancha existentes. Veremos de mano de quiénes se está introduciendo esta tecnología en el mercado y cual es la situación de WiMAX, especialmente en España. Mencionaremos algunos de los proyectos y pruebas se han realizado en nuestro país con WiMAX hasta el momento.

**\*Nota:** Aunque en un principio planificaron las pruebas de nuestra aplicación pensando en usar un acceso a Internet a través de WiMAX, finalmente debido a varios contratiempos no ha sido posible realizar dichas pruebas con la citada tecnología. Por tanto, las pruebas realizadas con Teletrófono han sido llevadas a cabo finalmente a través de un acceso Wi-Fi.

### 3.1. INTRODUCCIÓN A WiMAX.

WiMAX es la tecnología de radio de banda ancha que abarca los estándares 802.16 de IEEE (*Institute of Electrical and Electronics Engineers*) e HIPERMAN (*High Performance Radio Metropolitan Area Network*) de ETSI. WiMAX son las siglas de 'Worldwide Interoperability for Microwave Access' y es la marca que certifica que un producto está conforme con los estándares de acceso inalámbrico mencionados. Esta tecnología es promovida por el 'WiMAX Forum'.

WiMAX es un estándar de transmisión inalámbrica de datos que proporciona accesos concurrentes en áreas de hasta **50 kilómetros** de radio y a velocidades de hasta **70 Mbps**, utilizando tecnología que **no requiere visión directa** con las estaciones base. WiMax es un concepto parecido a Wi-Fi pero con mayor cobertura, con lo que puede dar servicio a un mayor número de usuarios, y mayor ancho de banda. Wi-Fi, fue diseñada para ambientes inalámbricos internos como una alternativa al cableado estructurado de redes y con capacidad sin línea de vista de muy pocos metros. WiMax, por el contrario, fue diseñado como una solución de última milla en redes metropolitanas (MAN- *Metropolitan Area Network*). WiMAX, por tanto, funcionaría de forma similar a WiFi pero a velocidades más altas, mayores distancias y para un mayor número de usuarios. (Fuente: Ibersystems)

Un sistema WiMAX tiene dos partes:

- Por un lado están las torres WiMAX, que dan cobertura de hasta 8.000 kilómetros cuadrados (unos 50 Km radio), según el tipo de señal transmitida. El elemento emisor o estación base (BS) estará formado por una antena y un equipo gestor de información.
- Por otro están los receptores, es decir, las tarjetas que conectamos a nuestro PC (*Personal Computer*), ordenador portátil, PDA (*Personal Digital Assistant*) y demás, para tener acceso. El elemento receptor o CPE (*Customer Premises Equipment*) será un pequeño módulo de interior o de exterior o una tarjeta PCMCIA acoplada a un ordenador.

Podemos encontrar dos formas de ofrecer señal:

- Cuando hay objetos que se interponen entre la antena y el receptor: **comunicación NLOS** (*Non Line Of Sight*). En este caso se opera con bajas frecuencias, entre los 2 y 11 Ghz, para así no sufrir interferencias por la presencia de objetos. Naturalmente esto conlleva que el ancho de banda disponible sea menor. Las antenas que ofrecen este servicio tienen una cobertura de 65 kilómetros cuadrados (unos 4,5 Km radio), aproximadamente como las de los teléfonos móviles.
- Cuando no hay nada que se interponga y hay contacto visual directo: **comunicación LOS** (*Line Of Sight*). En este caso se opera a muy altas frecuencias, del orden de 10

GHz hasta los 66 GHz, disponiendo de un gran ancho de banda. Además, las antenas que ofrezcan este servicio tendrán una cobertura de hasta 9.300 Km cuadrados (unos 55 Km radio). A frecuencias altas, la transmisión es más vulnerable a las condiciones climatológicas, pero se alcanzan mayores velocidades.

La mayor parte de usuarios, van a ser usuarios del primer tipo de servicio, el que opera a bajas frecuencias. En dicho servicio, a pesar de ser peor que el que se da cuando estamos en línea de vista, se va a notar mucha diferencia con el Wi-Fi más extendido (el 802.11-b/g) en dos aspectos fundamentales: la velocidad sube ahora hasta los 70 Mbps y la señal llega a ser válida hasta en 50 Km (con condiciones atmosféricas favorables). Finalmente, la velocidad real dependerá del número de usuarios conectados simultáneamente y de los requerimientos de ancho de banda de cada uno. (Fuente: Lamelas, 2006)

### **Redes WiMAX**

WiMAX utiliza una arquitectura punto a punto, punto-multipunto, o incluso de redes malladas, para proporcionar servicios de hasta un radio máximo de varios kilómetros, lo que depende de la frecuencia, la potencia emisora y la sensibilidad del receptor. En áreas con una alta densidad de población el radio de acción se encontrará limitado en función de esa densidad debido a la escasez de espectro disponible. El radio de acción y la capacidad NLOS hacen que esta tecnología sea enormemente atractiva en una gran variedad de entornos. En un principio esta tecnología fue prevista para proporcionar acceso de banda ancha inalámbrico en la última milla en redes de área metropolitana (MAN), y servicios iguales o algo mejores que los tradicionalmente ofrecidos por el cable, DSL o las líneas subcontratadas T1/E1. Sin embargo, el estándar 802.16 ayudará a proporcionar servicios a través de muchos segmentos de banda ancha como los referidos a continuación: enlaces *backhaul* móviles, banda ancha bajo demanda, acceso residencial a Internet a alta velocidad, pequeñas y medianas empresas, áreas desprovistas. (Fuente: IBM, 2006)

WiMAX se puede usar para enlaces de acceso MAN o incluso WAN (*Wide Area Network*). Destaca WiMAX por su capacidad como tecnología portadora, sobre la que se puede transportar IP (*Internet Protocol*), TDM (*Time-Division Multiplexing*), T1/E1 (jerarquías norteamericana y europea de PDH- *Plesiochronous Digital Hierarchy*), ATM (*Asynchronous Transfer Mode*), *Frame Relay* y voz, lo que la hace perfectamente adecuada para entornos de grandes redes corporativas de voz y datos así como para operadores de telecomunicaciones.

El cableado, como solución de última milla, representa altos costos de instalación que no siempre justifican su tendido hasta áreas rurales de difícil acceso. Llevar servicios ADSL (*Asymmetric Digital Subscriber Line*) a estas áreas no es costosamente efectivo para los operadores de telefonía. La tecnología celular presente, sólo permite la transferencia de archivos con calidad aceptable pero no permite la transmisión en tiempo real de aplicaciones multimedia. WiMAX podría solventar la carencia de acceso de banda ancha a las áreas suburbanas y rurales que las compañías del teléfono y cable todavía no ofrecen. (Fuente: Lamelas, 2006)

El acceso inalámbrico de banda ancha en la última milla puede ayudar a acelerar el uso y despliegue masivo de puntos de acceso (*hotspots*) 802.11 y la utilización de redes de área local inalámbricas en hogares y pequeñas oficinas, especialmente, como hemos dicho, en aquellas áreas que no se encuentran bien provistas por el cable o por DSL.

WiMAX permite a un proveedor de servicios inalámbrico ofrecer un servicio de velocidad comparable a una solución con hilos en cuestión de días, y con un coste significativamente más económico.

La tecnología 802.16 también permite al proveedor de servicios ofrecer conexión de alta velocidad “bajo demanda” de forma instantánea para acontecimientos puntuales incluyendo muestras comerciales que pueden generar cientos o miles de usuarios de *hotspots* 802.11. En estas aplicaciones, los operadores utilizan soluciones 802.16 para acceder a la red central. (Fuente: IBM, 2006)

Se considera que la gran oportunidad para WiMAX son los países en desarrollo, donde la necesidad de servicios de voz y datos se ve entorpecida por una pobre infraestructura de cable. En estos países resultaría una buena alternativa para el despliegue rápido de servicios. WiMAX competiría directamente con las infraestructuras basadas en redes de satélites, que son muy costosas y presentan una alta latencia. (Fuente: Pineda Marín)

WiMAX, gracias a sus características, puede resultar muy adecuado para unir puntos de acceso Wi-Fi a las redes de los operadores, sin necesidad de establecer un enlace fijo. El equipamiento Wi-Fi es relativamente barato pero un enlace E1 (PDH) o DSL (*Digital Subscriber Line*) resulta caro y a veces no se puede desplegar, por lo que la alternativa radio parece muy razonable. Para las empresas, es una alternativa a contemplar, ya que el coste puede ser hasta 10 veces menor que en el caso de emplear un enlace E1 o T1. WiMAX extiende el alcance de Wi-Fi y puede utilizarse como una seria alternativa o complemento a las redes 3G, según como se mire.

La robustez en el ancho de banda de la tecnología 802.16 la convierte en una excelente elección para enlaces *backhaul* en empresas comerciales *hotspots* así como aplicaciones *backhaul* punto a punto.

Son muchos los expertos que piensan que una red combinada de Wi-Fi e implementación WiMAX, ofrece una solución más eficiente en cuanto a costes que una implementación exclusiva de antena direccional Wi-Fi. WiMAX se usaría como *backhaul* de gran distancia y solución de última milla. Sin embargo, la solución deberá variar de acuerdo a los modelos de uso, el tiempo de implementación, la posición geográfica y la aplicación de red. En definitiva, lo que se quiere expresar es que es muy probable que las redes de área local Wi-Fi coexistan con WiMAX. (Fuente: Domínguez, 2008)

La instalación de estaciones base WiMAX es sencilla y económica, por lo que por los operadores móviles puede ser visto como una amenaza, pero también, es una manera fácil de extender sus redes y entrar en un nuevo negocio en el que ahora no están, lo que se presenta como una oportunidad. Los reyes del acceso a Internet por banda ancha son el ADSL y el cable, con WiFi para llevar la Red por el aire dentro de un espacio de unos centenares de metros y UMTS/HSDPA (*Universal Mobile Telecommunications System/High Speed Downlink Packet Access*) para hacerla completamente móvil. WiMax viene a trastocar un poco los planes de todas estas tecnologías, pues a todas les puede llegar a afectar de alguna manera. La instalación es mucho más barata que la del UMTS/HSDPA o las redes de cable; una pequeña inversión será suficiente para cubrir una ciudad entera con servicios de voz y datos sin necesidad de abrir zanjas. Algunos operadores de LMDS (*Local Multipoint Distribution System*) empezaron hace algunos años a considerar esta tecnología muy en serio y ya comenzaron a hacer despliegues de red, utilizando los elementos que han ido teniendo disponibles. (Fuente: Huidobro)

La batalla actual entre los proveedores de acceso a Internet está en la última milla: el bucle local o tramo del cable que llega hasta los hogares. El desarrollo de WiMAX podría acabar con el dominio del mercado del que disfrutaban los propietarios de las líneas que van desde las centralitas a cada domicilio (en España casi en exclusiva de Telefónica). Con esta nueva tecnología, cualquier proveedor podrá ofrecer acceso a Internet de banda ancha directamente a las casas, sin necesidad de tender una red de cable hasta cada hogar. Y, aunque WiMAX nació con el objetivo de cubrir la última milla, se prevé que también será capaz de ofrecer una alternativa a las conexiones por cable y ADSL. (Fuente: Domínguez, 2008)

En la figura 4.1 podemos ver algunos de los escenarios a los que puede dar solución la tecnología WiMAX:



Figura 3.1. Escenarios WiMAX.  
(Fuente: GEARFUSE, 2008)

### **WiMAX Forum**

El WiMAX Forum es un consorcio de empresas (hoy en día más de 500) dedicadas a diseñar los parámetros y estándares de esta tecnología, y a estudiar, analizar y probar los desarrollos implementados. Es una organización no lucrativa formada en Junio de 2001 por unas 255 compañías líderes en el desarrollo de tecnología en telecomunicaciones para: armonizar estándares, promocionar la tecnología y promover y certificar la compatibilidad e interoperabilidad de las redes inalámbricas de banda ancha. Sus dos miembros más representativos son Intel y Nokia, pero también están presentes CISCO Systems, SR Telecom, France Telecom...

Aquel equipo certificado por el Forum WiMAX está diseñado y es configurable para varios escenarios de acceso inalámbrico de banda ancha. Estos escenarios incluyen desde la posibilidad de largo alcance (en teoría 50 kms) con baja densidad de población y condiciones LOS hasta despliegues de un menor alcance, con condiciones NLOS y entorno urbano densamente poblado. Los servicios pueden ser fijos, portátiles o móviles, o una combinación de los mismos. (Para más información: <http://www.wimaxforum.org/>)



### 3.2. ESTÁNDARES WiMAX.

WiMAX es la marca que certifica que un producto está conforme con los estándares de acceso inalámbrico IEEE 802.16. Desde su concepción en los años noventa, el estándar IEEE 802.16 ha tenido como objetivo primordial el de proveer diferentes tipos de servicios sobre una red de acceso inalámbrico de alta transferencia de bits. Para ello define unas especificaciones de interfaz incluyendo la MAC (*Medium Access Control*), y la PHY (Capa Física).

El primero de los estándares se publicó en 2001 y se denominó 802.16. Más tarde aparecieron las versiones IEEE 802.16b c y d, que son en realidad enmiendas del estándar base. Después se publicó el estándar 802.16-2004, que fue diseñado para servir como una tecnología de reemplazo del DSL inalámbrico, para proveer un acceso básico de voz y banda ancha en áreas desprovistas y para proveer una solución para el *backhaul* inalámbrico entre puntos de acceso Wi-Fi o en redes celulares. En 2005 apareció un nuevo estándar, el IEEE 802.16e. Este último es una revisión de la especificación del 2004 pero diseñado para ofrecer una característica clave de la que carecía dicha especificación: portabilidad. El estándar 802.16e requiere una nueva solución de hardware/software ya que no es compatible con el anterior 802.16-2004, lo cual no es algo bueno para los operadores que estaban planeando desplegar el 802.16-2004 y luego ascender al 802.16e. Desde entonces, los trabajos de estandarización en curso, hasta el año 2009, han sido de cariz más técnico, para corregir o aumentar funcionalidades ya existentes. En 2009, se ha publicado un estándar, el IEEE 802.16-2009. Este estándar ha pasado a ser el único estándar vigente en la actualidad. Incluye tanto aplicaciones fijas como móviles, e incluso aplicaciones de *backhaul* en alta frecuencia. Deja obsoletas las versiones 802.16-2004, 802.16e, 802.16f y 802.16g. (Fuente: IEEE Standards Association)

	<b>802.16</b>	<b>802.16a/REVd</b>	<b>802.16e</b>
<b>Publicado</b>	Diciembre 2001	Enero 2003- Septiembre 2004	Diciembre 2005
<b>Espectro</b>	10 - 66 GHz	< 11 GHz	< 6 GHz
<b>Funcionamiento</b>	Sólo con visión directa (LOS)	Sin visión directa (NLOS)	Sin visión directa (NLOS)
<b>Tasa de bit</b>	32 - 134 Mbps con canales de 28 MHz	Hasta 75 Mbps con canales de hasta 20 MHz	Hasta 15 Mbps con canales de 5 MHz
<b>Modulación</b>	QPSK, 16QAM y 64 QAM	OFDM 256, OFDMA 64 QAM, 16QAM, QPSK	Igual que 802.16a
<b>Movilidad</b>	Sistema fijo	Sistema fijo	Movilidad pedestre
<b>Anchos de banda del canal</b>	20, 25 y 28 MHz	Seleccionables entre 1,25 y 20 MHz	Igual que 802.16a.
<b>Radio de celda típico</b>	2 - 5 km aprox.	5 - 10 km aprox. (alcance máximo de unos 50 km)	2 - 5 km aprox.

Figura 3.2. Tabla comparativa de estándares 802.16.

(Fuente: [http://www.tutorialspoint.com/images/wimax\\_ieee\\_standards.gif](http://www.tutorialspoint.com/images/wimax_ieee_standards.gif))

### 3.3. CARACTERÍSTICAS GENERALES WiMAX.

Uno de los principales obstáculos a superar para acelerar el acceso inalámbrico de banda ancha son los costes. Aunque el coste total del despliegue incluye muchos factores (licencias, torres, gastos de *backhaul*), la mayor parte de este coste se la llevan los equipos y este es el centro de atención de los proveedores de servicios y los fabricantes del Forum WiMAX en este aspecto. La armonización global, o el reparto uniforme del espectro a lo largo del Globo, es crucial para abaratar los costes de los equipos.

El WiMAX Forum aboga por que los gobiernos permanezcan neutrales en el reparto del espectro. Las bandas del espectro se deben repartir de manera que permita a cada licencia ofrecer los servicios y tecnologías más apropiados para su mercado. Todo ello siempre y cuando las soluciones permanezcan acordes a las normas reguladoras, es decir, que puedan soportar compatibilidad de servicios y se comporten de una manera justa y segura. El Forum WiMAX tiene el cometido de crear normas que aseguren beneficios económicos y una utilización eficiente del espectro.

Las bandas iniciales de interés fueron:

- Banda sin licencia de 5 GHz: El rango de frecuencias de interés incluye las bandas entre los 5.25 y 5.85 GHz. Debido a que en la mayoría de países el espectro sin licencia se puede usar libremente, esta banda es estratégica para despliegues totales, desde el principio, en áreas desabastecidas de baja densidad de población. A diferencia de WiFi que apunta a aplicaciones para redes de área local (en exteriores), WiMAX apunta a aplicaciones en áreas metropolitanas (interiores y exteriores) lo cual implica que son necesarios mayores niveles de potencia. Muchos países permiten niveles de salida de potencia aceptables en la parte más alta de la banda de 5GHz (5.725-5.850 GHz), lo que las convierte en frecuencias atractivas para WiMAX.
- Banda con licencia de 3.5 GHz: El reparto principal con licencia para el acceso inalámbrico de banda ancha se encuentra entre los 3.4 y 3.6 GHz. El objetivo del Forum WiMAX en estas bandas será minimizar los requisitos técnicos y reguladores necesarios, que puedan inhibir un desarrollo global del mercado.
- Banda con licencia de 2.5 GHz: Las bandas entre 2.5 y 2.69 GHz se han repartido en los EEUU, México, Brasil y algunos países del sureste asiático. También es frecuente la banda de 2.3 GHz en el sureste asiático (incluyendo Australia, Nueva Zelanda y Corea del Sur), que el Forum WiMAX espera sustituir por la banda de 2.5 GHz.
- Nuevas bandas de interés en baja frecuencia (< 1GHz): Sabemos que las ondas de radio se propagarán más lejos cuanto más baja es la frecuencia, creando una relación directa entre el número de estaciones base necesarias para cubrir un área de servicio dada. El Forum WiMAX desarrolla estudios compartidos para demostrar cómo los equipos certificados por el WiMAX Forum pueden operar de manera compatible en bandas adyacentes a las de retransmisión de televisión y otras aplicaciones en bandas de bajas frecuencias. (Fuente: IBM, 2006)

A continuación se muestra una lista en la que aparecen algunos países o zonas del mundo con las frecuencias que se utilizan o utilizará WiMAX:

- Canadá, América central y Sudamérica: 2.5, 3.5 y 5 GHz.
- Estados Unidos: 2.5 y 5 GHz.
- Europa y África: 3.5 y 5 GHz.
- Rusia: 2.5, 3.5 y 2.3 GHz.
- Asia y Pacífico: 2.3, 3.3, 3.5 y 5 GHz.

WiMAX maneja anchos de banda variables para adaptarse a las diferentes bandas y características de cada país. Trabaja con anchos de canal que van desde los 1,25 MHz hasta los 28 MHz. Otro aspecto importante del estándar 802.16x es que define un nivel MAC (*Media Access Layer*) que soporta múltiples enlaces físicos (PHY). Esto es esencial para que los fabricantes de equipos puedan diferenciar sus productos y ofrecer soluciones adaptadas a diferentes entornos de uso. Los tipos de capa física que WiMAX soporta son: WirelessMan-SC, WirelessMan-SCa, WirelessMan-OFDM y WirelessMan-OFDMA. (Fuente: Huidobro)

## MODULACIÓN

WiMAX incluye mecanismos de modulación adaptativa, mediante los cuales la estación base y el equipo de usuario se conectan utilizando la mejor de las modulaciones posibles, en función de las características del enlace radio. La utilización de modulación adaptativa permite, en un sistema inalámbrico, elegir la modulación de orden más alto dependiendo de las condiciones del canal. Los diferentes órdenes de modulación, cuanto más altos son, permiten enviar más bits por símbolo y por lo tanto lograr mejores rendimientos o mejores eficiencias espectrales. Sin embargo, es necesario hacer notar que cuando se utiliza una técnica de modulación como 64-QAM, se necesitan mejores relaciones señal a ruido (SNRs) para superar las interferencias y mantener una cierta tasa de error de bit (BER).

A medida que se aumenta el alcance, se utilizarán modulaciones más bajas (menos finas), pero cuando se está cerca de las estaciones base es posible la utilización de modulaciones de órdenes mayores como QAM para incrementar el rendimiento. Además, la modulación adaptativa permite al sistema superar el *fading* y otras interferencias. Tanto QAM como QPSK son técnicas de modulación utilizadas habitualmente por tecnologías inalámbricas. (Fuente: IBM, 2006)

## TÉCNICAS DE RADIO

WiMAX utiliza avanzadas técnicas de radio como las ‘*smart antenas*’, la diversidad en transmisión y MIMO (*Multiple Input Multiple Output*).

1. WiMAX incorpora soporte para tecnologías ‘*smart antenas*’ (antenas inteligentes), propias de las redes celulares de tercera generación, que mejoran la eficiencia espectral (llegando a conseguir 5 bps/Hz, el doble que en 802.11a) y la cobertura.

Los sistemas de antena adaptada (AAS) son opcionales dentro del estándar 802.16. Estas antenas inteligentes emiten un haz muy estrecho que se puede ir moviendo, electrónicamente para enfocar siempre al receptor, con lo que se evitan las interferencias entre canales adyacentes y se consume menos potencia al ser un haz más concentrado. Mejora la reutilización de frecuencias. También se puede llamar *beamforming* o conformación de haz.

2. La diversidad es una característica opcional de WiMAX. Los esquemas de diversidad se utilizan para aprovechar el multitrayecto y la reflexión de las señales que se producen bajo condiciones NLOS. Utilizando varias antenas (transmisoras y/o receptoras) es posible reducir el *fading*, las pérdidas por trayecto y las interferencias. Se trata de enviar/recibir lo mismo a través de varias antenas. Mejora la fiabilidad. La diversidad resulta ser una herramienta útil y efectiva para superar los problemas que nos plantea la propagación NLOS.

3. MIMO consiste en la utilización de varias antenas tanto en transmisión como en recepción. Utiliza muchas antenas receptoras y/o transmisoras para la multiplexación espacial. Cada antena podría transmitir datos diferentes que podrían ser enviados por diferentes caminos y ser decodificados en el receptor. MIMO logra aumentar la capacidad de una conexión inalámbrica sin necesidad de asignar más ancho de banda ni aumentar la potencia.

Si se ponen múltiples antenas en transmisión y en recepción podremos aumentar el ancho de banda de una forma lineal con el número de antenas, siempre que se garantice suficiente propagación multitrayecto.

(Fuente: IBM, 2006)

### **QoS (QUALITY OF SERVICE)**

WiMAX soporta varios cientos de usuarios por canal, con un gran ancho de banda y es adecuada tanto para tráfico continuo como a ráfagas, siendo independiente de protocolo; así, transporta IP, Ethernet, ATM etc. y soporta múltiples servicios simultáneamente ofreciendo Calidad de Servicio (QoS) a partir de 802.16e, por lo cual resulta adecuado para voz sobre IP (VoIP), datos y vídeo. La QoS también ha sido diseñada dentro del estándar para permitir servicios que requieren baja latencia, como voz y vídeo. Además WiMAX facilita varios niveles de servicio para poder dar diferentes velocidades de datos dependiendo del contrato con el suscriptor.

**Tipos de Servicio:** Se refiere a cómo se planifica la transmisión desde la estación base hacia los usuarios (se puede hacer garantizando o no calidad de servicio).

- *Unsolicited Grant Service* (UGS): Soporta paquetes de datos de longitud fija, que requieren tiempo real (transmitidos periódicamente). Ej: VoIP sin supresión de silencios, ATM CBR (*Constant Bit Rate*) y E1/T1 sobre ATM.
- *Real Time Polling Service* (rtPS): Soporta paquetes de datos de longitud variable, que requieren tiempo real (transmitidos periódicamente). Ej: VoIP, *streaming* de vídeo y audio.
- *Non Real Time Polling Service* (nrtPS): Soporta paquetes de datos de longitud variable, tolerantes al retardo, con una velocidad de transmisión mínima. Ej: FTP.
- *Best Effort* (BE): Cadenas de datos para los que no existe un nivel de servicio mínimo requerido, no se ofrecen garantías de tratamiento ni retardo y se pueden enviar en el “espacio disponible” (según las oportunidades de transmisión que permita la carga de la red).

### **Modos de operación:**

- *Grant Per Connection* (GPC): La BS garantiza el ancho de banda para cada conexión y la SS utiliza este ancho de banda sólo para esta conexión.

- *Grant Per Subscriber Station (GPSS)*: La BS garantiza un ancho de banda general que será utilizado por todas las conexiones pertenecientes a la SS. La SS debe administrar la QoS y asegurar el reparto del ancho de banda entre todas las conexiones que lo piden, esto pide que trabaje más a nivel de la SS.

(Fuente: IBM, 2006)

## **CORRECCIÓN DE ERRORES**

Las técnicas de corrección de errores se han incorporado a WiMAX para reducir la tasa de error de la señal del sistema a unos niveles admisibles. Para detectar y corregir errores y mejorar el rendimiento se utiliza:

- Strong Reed Solomon FEC (*Forward Error Correction*)
- Codificación Convolutiva
- Algoritmos de Interleaving

Las técnicas de corrección de errores, que son muy robustas, ayudan a recuperar tramas erróneas que se podrían haber perdido debido al *fading* selectivo en frecuencia o a una ráfaga de errores. Los errores que FEC no puede corregir se intentan corregir con ARQ, reenviando la información errónea. Esto mejora significativamente la tasa de error de bit (BER). (Fuente: IBM, 2006)

## **SEGURIDAD**

La seguridad en 802.16 se implementa como una subcapa de privacidad al final del protocolo de capas internas de la capa MAC. Su finalidad es proveer de control de acceso y confidencialidad al enlace de datos.

Encriptando los enlaces entre la BS y las SS, WiMAX proporciona a los usuarios privacidad y seguridad en la interfaz inalámbrica de banda ancha. WiMAX ha incorporado también soporte VLAN que provee protección para los datos que transmiten varios usuarios en la misma BS.

Por último vamos a enumerar brevemente las tecnologías de seguridad más importantes que soporta WiMAX:

- **Protocolo de gestión de claves:** PKMv2 (*Privacy and Key Management Protocol version 2*) es la base para la seguridad en WiMAX, tal y como se define en 802.16e. Este protocolo controla la seguridad MAC usando mensajes PKM-REQ/RSP.

- **Autenticación de dispositivo/usuario:** La autenticación de dispositivos y usuarios se lleva a cabo usando el protocolo EAP que provee soporte a credenciales.

- **Encriptación de tráfico:** el protocolo AES-CCM es el cifrado usado para proteger los datos de usuario. La autenticación EAP se encarga de generar las claves necesarias para construir el sistema de cifrado.

- **Protección de los mensajes de control:** Los datos de control se protegen usando esquemas 3AES basados en CMAC (*Cipher-based Message Authentication Code*), o MD5 basados en HMAC (*Hash-based Message Authentication Code*).

- **Soporte rápido de *handover*:** Se utiliza un esquema '*3-way Handshake*' que optimiza los mecanismos de re-autenticación para ofrecer un rápido traspaso entre celdas.

(Fuente: IBM, 2006)

### 3.4. WiMAX FRENTE A OTRAS TECNOLOGÍAS.

La creciente demanda de datos y contenidos multimedia ha provocado un incremento del desarrollo y la evolución de las redes inalámbricas. Este crecimiento ha sido impulsado en gran medida por el éxito del *streaming* de datos multimedia en Internet. Además la disponibilidad “en cualquier lugar” de tecnología digital multimedia ha visto una afluencia repentina de tecnologías de red, algunas complementarias a las ya existentes, y algunas sustitutivas, trayendo consigo un aumento en el rendimiento (throughput).

En este apartado vamos a revisar brevemente cual es la posición de WiMAX respecto a otras tecnologías de red inalámbricas que ya estén implantadas en el mercado.

En un principio se podría pensar en que WiMAX es una directa competencia a WiFi, pero veremos que no es así. Sin embargo, otras tecnologías como HSPDA (*High-Speed Downlink Packet Access*), que no de había planteado como competidor, o la reciente tecnología 802.20, sí se podrían interponer a los planes de expansión de WiMAX.

En primer lugar, vamos a presentar, en la Figura 3.3, una tabla comparativa de algunas de las características principales tecnologías inalámbricas. Vemos que el estándar 802.16 puede alcanzar una velocidad de comunicación de más de 100 Mbps (en un canal con un ancho de banda de 28 MHz, en la banda de 10 a 66 GHz). (Fuente: Huidobro)

	<b>WiMAX 802.16</b>	<b>Wi-Fi 802.11g</b>	<b>Mobile-Fi 802.20</b>	<b>HSDPA</b>
Velocidad (máxima teórica)	124 Mbps	54 Mbps	16 Mbps	14 Mbps
Cobertura	5-50 km	10m-300 m	20 km	1 km- 20 km
Licencia	Si/No	No	Si	Si
Ventajas	Velocidad y Alcance	Velocidad y Precio	Velocidad y Movilidad	Rango y Movilidad
Desventajas	Aún en despliegue	Bajo alcance	Precio alto	Poca velocidad. Alto precio

Figura 3.3. Cuadro comparativo WiMAX vs. otras tecnologías inalámbricas.

(Fuente: Huidobro)

#### 3.4.1. WiMAX vs. Wi-Fi.

WiMAX no ha sido diseñado para competir con Wi-Fi sino más bien para complementar a Wi-Fi en aquellas carencias que éste presenta.

La primera norma inalámbrica (802.11) fue desarrollada como una alternativa al cableado estructurado de redes LAN. Para entender mejor las aplicaciones para las cuales Wi-Fi fue diseñado hay que imaginar una red Ethernet dentro de una oficina durante los años noventa. El requerimiento era una red dentro de una oficina. Wi-Fi fue

diseñado para ambientes inalámbricos internos y las capacidades sin línea de vista (NLOS) son posibles únicamente para unos pocos metros. A pesar de este diseño y de todas las limitaciones, había muchos proveedores de Internet (ISP) que implementaban radios Wi-Fi para servicio de ‘última milla’. Debido al diseño de Wi-Fi, los servicios en estas redes eran bastante limitados. En los últimos años hemos visto mucho desarrollo en Wi-Fi y Ethernet para adaptarse a los cambios en las redes de datos. Esto incluye mejor seguridad (encriptación), redes virtuales (VLAN), y soporte básico para servicios de voz (QoS). (Fuente: Lamelas, 2006)

· Características Wi-Fi:

- Optimizado para usuarios en un radio de 10 metros. Cobertura máxima de 100 m.
- Para una mayor cobertura hay que añadir puntos de acceso o antenas de alta ganancia.
- Optimizado para interiores.
- Las capacidades sin línea de vista (NLOS) son posibles únicamente para unos pocos metros.
- Aplicación LAN. El número de usuarios puede variar entre uno y varias decenas con un suscriptor CPE.
- Para 20 Mhz el ancho de banda del canal es fijo.
- 2,7 bps/Hz es la eficiencia espectral máxima. Se consiguen 54 Mbps en la versión 802.11g en un canal de 20 Mhz.
- QoS soportada sólo a partir de 802.11e.
- Usa 64- OFDM.

· Características WiMAX:

- Optimizado para un tamaño típico de celda de 12-15 Km (LOS), 1-2 Km (NLOS).
- Supera los 50 Km. de alcance.
- Sin problemas en escenarios NLOS (en la banda 2-11 GHz).
- Optimizado para espacios exteriores.
- Soporte eficiente para cientos de SSs con un número limitado de usuarios.
- Es estándar soporta avanzadas técnicas de antena (*smart* antenas) y redes malladas.
- El ancho de banda del canal es flexible desde los 1,5 MHz hasta los 20 MHz tanto para bandas de frecuencia con licencia como para las exentas de licencia.
- Re-utilización de frecuencias.
- Permite planificación de celdas para proveedores de servicio comerciales.
- 3,8 bps/Hz es la eficiencia espectral máxima. Más de 75 Mbps en un canal de 20 MHz.
- 5 bps/Hz es la eficiencia espectral máxima. 100 Mbps en un canal de 20 MHz.
- QoS para voz y vídeo.
- Tolera 10 veces más “*multi-path delay spread*” que Wi-Fi.
- Usa 256 OFDM: es una solución robusta para operar en condiciones N-LOS a distancias de varios kilómetros.
- WiMAX facilita varios niveles de servicio (MIR/CIR) para poder dar diferentes velocidades de datos dependiendo del contrato con el suscriptor.

Como ya hemos mencionado en otras ocasiones, WiMAX y Wi-Fi son soluciones complementarias para dos aplicaciones bastante diferentes. WiMAX fue diseñado para redes metropolitanas (MAN), también conocido como “última milla”. Wi-Fi fue diseñada para redes locales (LAN), también conocido como “distribución en sitio”.

Sin embargo, recientemente se ha publicado la versión IEEE 802.11n-2009, que ha mejorado notablemente la velocidad de datos pasando de 54 Mbps a un máximo de 600Mbps, así como aumenta unas cinco veces el alcance. Este estándar se basa en los anteriores de 802.11 y añade la técnica MIMO (*multiple input multiple output*) y canales de 40 MHz. Si recordamos, MIMO también es una tecnología usada por WiMAX. Dos importantes beneficios que aporta a 802.11n son la diversidad de antena y la multiplexación espacial. (Fuente: Broadcom, 2006)

La altísima tasa de transmisión que ofrece esta tecnología (en realidad con los dispositivos actuales se suele alcanzar un máximo de 300 Mbps) podría hacernos pensar que WiMAX ya no tiene nada que hacer como red inalámbrica de banda ancha. Pero recordemos, que 802.11n trabaja en las bandas de frecuencia de 2.4 GHz y 5GHz que no requieren licencia, mientras que WiMAX puede trabajar en espectros licenciados. Este es el principal inconveniente que se le plantea al 802.11n para competir directamente con WiMAX: las interferencias son un gran problema para 802.11n si estamos planteándonos aprovechar esta tecnología en espacios al aire libre y además, nunca podrá tener un alcance similar al de WiMAX porque su potencia está limitada (al trabajar en espectro sin licencia).

### **3.4.2. WiMAX vs. HSDPA.**

La competencia entre HSDPA y WiMAX es inevitable, al menos por ahora, aunque sólo sea por la sencilla razón de que los objetivos actuales de ambos se cuelan en el campo tradicional de trabajo del otro. HSDPA se está moviendo en la tecnología de datos, el trampolín de la tecnología WiMAX, mientras que WiMAX está extendiéndose a la tecnología de voz, el precursor de la evolución de la tecnología HSDPA.

HSDPA es una característica de UMTS (*Universal Mobile Telecommunications System*) Release 5 que amplía las características de la tercera generación (3G) en el enlace descendente con una capacidad máxima teórica de 14Mbps. Se trata de un canal de transmisión compartido, con un porcentaje (normalmente 80%) del total de los recursos del enlace radio descendente compartido dinámicamente por los usuarios. (Fuente: 3G AMERICAS, 2004)

#### **Comparación técnica**

En HSDPA, el nuevo canal de transporte de alta velocidad tiene un factor de ensanchado fijo de 16 códigos de los cuales uno se reserva para señalización. Esto significa que los usuarios que estén experimentando una buena relación señal a ruido pueden usar una codificación 16QAM. Sin embargo, como todos los usuarios están multiplexados en el tiempo en el mismo espectro, independientemente de su situación geográfica, el efecto combinado de la interferencia y el multitrayecto lo llevarán a tener que usar un esquema de codificación más bajo, QPSK. Sin embargo se puede utilizar una técnica que permite al equipo de usuario transmitir en aquel canal que esté en buenas condiciones o evitar aquellos canales que estén experimentando grandes pérdidas. (Fuente: 3G AMERICAS, 2004)

Por el contrario, con OFDMA (técnica usada por WiMAX) sólo puede seleccionar entre 16 y 64 QAM dependiendo de las condiciones de canal. Al utilizar un esquema de modulación más alto se obtiene una mayor subcanalización. Además WiMAX tiene una distribución dinámica de canales para el reparto de la eficiencia espectral, expandiendo



el ancho de banda del enlace de subida o de bajada dependiendo del tráfico que sea menor.

HSDPA usa CDMA (*Code Division Multiple Access*), mientras que WiMAX usa SOFDMA. En las tecnologías de acceso de espectro ensanchado cuando hay un fuerte incremento del multitrayecto en la señal de radio, la capacidad del sistema se ve reducida. Las tecnologías *Direct Sequence Spread Spectrum* (DSSS) como CDMA son muy susceptibles al *fading* y al *shadowing* y los símbolos recibidos fuera de tiempo causan ISI (interferencia entre símbolos).

Como WiMAX va a utilizar frecuencias portadoras mayores, la atenuación asociada será también mayor. Los sistemas OFDMA, sin embargo, sufren en mayor modo el efecto Doppler, con el que las subportadoras dejan de ser ortogonales y se causa ICI (*Inter-Carrier Interference*).

La capacidad de una red inalámbrica es normalmente función de su ancho de banda y su eficiencia espectral. HSDPA hereda de los sistemas de circuitos conmutados (de la telefonía) y la escalabilidad del espectro no es precisamente una ventaja presente en estos sistemas. Sin embargo, WiMAX proporciona un ancho de banda escalable.

HSDPA, como sabemos, proviene de la telefonía móvil, por lo que tiene una ventaja enorme respecto a WiMAX en términos de movilidad. Esto se puede notar, en lo que se refiere a traspasos entre celdas. Mientras HSDPA utiliza un *handover* ‘duro’, WiMAX utiliza tres: ‘duro’, conmutación rápida de la estación base y traspaso de ‘macro-diversidad’. Dado que WiMAX tiene más celdas para una misma superficie que HSDPA, habrá más posibilidades de que se produzca un error al haber más tipos de *handover* y un mayor número de éstos a realizar. Esto hace que WiMAX sea solamente muy eficiente en situaciones de ‘portabilidad’ o ‘nómadas’.

HSDPA y WiMAX proporcionan un conjunto de herramientas para soportar multitud de aplicaciones como por ejemplo servicios en tiempo real o de *streaming*. Sin embargo el estándar 802.16 especifica unos requerimientos de QoS más seguros y robustos.

WiMAX y HSDPA tienen la opción de implementar varias tecnologías de antena que mejoran la capacidad y cobertura. Sin embargo, estas tecnologías son más numerosas y están mejor implementadas en WiMAX que en HSDPA.

HSDPA se implementa en las redes 3G, a veces con tan sólo una actualización de software o como máximo unas tarjetas de hardware (módems) en el RNC (*Radio Network Controller*) y Nodo B. El gasto de capital es, por tanto, muy bajo. En la actualidad, apenas hay redes WiMAX comerciales en funcionamiento. Esto podría atribuirse al hecho de que los gastos de capital para la misma cobertura que HSDPA es mucho mayor, ya que los operadores tendrían que empezar de cero y, con poca demanda, el proyecto podría no ser comercialmente viables. (Fuente: iXBT LABS, 2006)

### **Comparación de la situación de ambas tecnologías**

#### **•Acerca de HSDPA:**

- Prácticamente todos los operadores 2G han seguido este camino.
- Tecnología estandarizada y con un ecosistema muy robusto, debido al apoyo de un gran número de empresas.
- Velocidades de transmisión en campo más que aceptables.
- La tecnología ya ha sido lanzada, demostrando su cierto grado de madurez.
- Operadores dispuestos a compartir la red para reducir costos de implementación.

- Acceso rápido a servicios de roaming de voz y datos, gracias a la compatibilidad de los dispositivos con GSM y UMTS/WCDMA.
- Amplia cobertura del servicio apoyado en la compatibilidad hacia atrás con UMTS y GSM.

• **Acerca de WiMAX:**

- Apoyo de operadores importantes (Sprint, Telecom Italia, Telmex) y fabricantes (Nokia, Motorola, Samsung, etc.)
- Tecnología muy útil para cubrir zonas con poca o cero penetración de infraestructura fija.
- Precio del espectro relativamente bajo (quizás reflejo de un modelo de negocio más incierto). Se paga más por el espectro en mercados con baja penetración de banda ancha fija.

**Conclusiones**

En definitiva, la provisión de servicios de banda ancha a través de redes inalámbricas tiene que conseguir principalmente dos cosas: movilidad y un alto rendimiento. En este sentido WiMAX y HSDPA parecen estar en extremos opuestos del problema. HSDPA puede ofrecer la movilidad total, y además tiene la ventaja de ser una red actualmente en funcionamiento, mientras que WiMAX puede garantizar mayores tasas de datos, pero parece no alcanzar todavía una movilidad satisfactoria. Lo ideal sería, por tanto, una red heterogénea donde la movilidad de HSDPA y el mayor rendimiento de la tecnología WiMAX sean explotados para lograr el éxito de la banda ancha inalámbrica móvil.

Sin embargo es de prever que HSDPA domine el sector masivo de acceso a banda ancha móvil por tres factores principales: los operadores ya están comprometidos con la tecnología debido a su infraestructura existente (GSM/UMTS), el precio de los dispositivos es bajo gracias a las economías de escala y HSDPA cuenta con un robusto servicio básico: voz. (Fuente: TeleSemana)

Por otro lado, WiMAX, podría ocupar un espacio importante en el sector de las telecomunicaciones especialmente en mercados emergentes para cubrir zonas con poca o cero infraestructura física, o en mercados con baja densidad de redes de telecomunicaciones. WiMAX también podría ser un gran aliado para las redes celulares, pero sin olvidar que estas tecnologías las desarrollan operadores diferentes.

No podemos pasar por alto la importancia de la voz. WiMAX hace más énfasis en los servicios de datos. Mientras que HSDPA tiene mayor robustez para la oferta de servicios de voz y datos, dada la infraestructura de legado existente (GSM). Además habrá que prestar gran atención a la regulación para que los operadores WiMAX móvil puedan ofrecer servicios de telefonía; los actuales operadores de telefonía podrían ejercer una gran presión en este tema. Sin duda, HSDPA se presenta como serio rival de WiMAX.

**3.4.3. WiMAX vs. Mobile-Fi.**

IEEE 802.20 o *Mobile Broadband Wireless Access* (MBWA) es un estándar de IEEE para posibilitar el desarrollo en todo el mundo de redes inalámbricas de acceso móvil de banda ancha con el objetivo de que fuera un modo de acceso asequible, interoperable y con la característica de “permanentemente conectado”. Cabe destacar que ha sido diseñado desde un principio para el uso intensivo de IP y en particular orientado a la

VoIP (*Voice Over Internet Protocol*) y aplicaciones IP. Mobile-Fi fue diseñada como una tecnología para proveer servicios de alta movilidad. Soporta movilidad hasta a 250 Km/h y proporcionará anchos de banda simétricos entre 1 y 4 Mbps en bandas de frecuencia reguladas por debajo de los 3.5 GHz, a distancias de la estación base de unos 15 Km. Las tecnologías que usa son OFDM, MIMO y *beam-forming*. Estas dos últimas son también utilizadas por WiMAX. (Fuente: GAPTEL, 2005)

802.16 y 802.20 podrían parecer iguales pero difieren en varios aspectos. Para empezar, ambos ofrecen servicios de datos simétricos, pero 802.16 proporciona de modo fijo una alta velocidad inalámbrica con un servicio añadido de movilidad, mientras que 802.20 proporciona una transmisión veloz con total movilidad. Las velocidades de transmisión en entornos móviles para 802.16 suelen ir de 10 a 50 Mbps y para 802.20 es como máximo de 16 Mbps. Ambas soportan servicios de datos con bajo retardo, y en el caso de 802.16 también soporta servicios de voz en tiempo real. A diferencia de 802.16, Mobile-Fi ofrece una movilidad global y soporte de *roaming*.

802.16 usa bandas de frecuencias licenciadas de 2-6 GHz mientras 802.20 trabaja también con bandas con licencia, pero por debajo de los 3.5 GHz. 802.16 soporta una velocidad de usuario de 60 Km/h y 802.20 de más de 250 Km/h. (Fuente: IEEE, 2003)

### **Conclusiones**

El objetivo principal de IEEE 802.16 es proporcionar servicios de voz y datos para usuarios tanto fijos como móviles, mientras que el de 802.20 es ofrecer conexiones móviles de banda ancha a usuarios móviles. El estándar 802.20 sirve a los usuarios un alto ancho de banda, bajo retardo y total movilidad. El estándar 802.20 usa un arquitectura de capas MAC y PHY mejorada y empieza desde cero, mientras que 802.16 usa versiones modificadas de las originales capas MAC y PHY que implementaban un acceso inalámbrico de banda ancha fijo. Pero, mejorar sistemas inalámbricos fijos existentes para lograr que estos lleguen a soportar una total movilidad no es posible, en tanto en cuanto el soporte de movilidad es un factor crítico del sistema que va más allá de los traspasos entre celdas y del *fading* multitrayecto. Sin embargo, el mayor alcance y *throughput* de WiMAX lo harán, probablemente imponerse ante Mobile-Fi.

### 3.5. LICENCIAS Y OPERADORES.

Tal y como se mencionó en apartados anteriores, en España, WiMAX se desarrolla en las bandas de frecuencia en torno a 3,5 GHz (banda que requiere licencia) y 5 GHz (banda sin licencia). En las bandas de frecuencia que no requieren licencia, existen restricciones de potencia (EIRP- *Effective Isotropic Radiated Power*) que afectarán a la capacidad de los sistemas de proveer servicios viables en largas distancias. Por ello las compañías que deseaban dedicarse de lleno a la implementación de WiMAX buscaban hacerse con una de las frecuencias en la banda en torno a 3,5 GHz.

Al ser necesario el uso privado por parte del espectro radioeléctrico, el número de licencias para explotar la banda de 3,5 GHz debía ser limitado y otorgarse mediante concesión por concurso y no por simple procedimiento de autorización.

Sin embargo, estas licencias no estaban desde el inicio pensadas para desarrollar sobre ellas un servicio WiMAX, sino que eran licencias para operar LMDS (*Local Multipoint Distribution Service*). LMDS, está definido en el estándar IEEE 802.16.1, es, por tanto, de la misma familia de estándares que WiMAX, IEEE 802.16. LMDS es un sistema de comunicación de punto a multipunto que utiliza ondas radioeléctricas a altas frecuencias. Debido a las altas frecuencias y al amplio margen de operación, es posible conseguir un gran ancho de banda de comunicaciones, con velocidades de acceso que pueden alcanzar los 8 Mbps. En España, el servicio LMDS se ofrecía en las frecuencias de 3,5 GHz ó 26 GHz. Proporcionaba alcances de 5-10 Km y velocidades de hasta 2 Mbps.

El Ministerio de Fomento, siguiendo la legislación aplicable a este tipo de licencias individuales, abrió un periodo de información pública que se extendió hasta mediados de septiembre de 1999. El objeto de este periodo de consulta era conseguir una estimación del número de grupos interesados en solicitar tres licencias en la banda de los 26 GHz y otras tres en la banda de los 3,5 GHz. Como el número de compañías interesadas superaba ampliamente la oferta disponible, el Gobierno tenía que convocar concursos. El 9 de octubre de 1999 se publicaron dos concursos, uno para la banda de los 26 GHz con tres licencias y otro para la de 3,5 GHz con otras tres. Hubo siete licitadores en el concurso de 3,5 GHz y diez en el de 26 GHz.

La adjudicación de las seis licencias fue hecha pública el 10 de marzo de 2000 en medio de una fuerte polémica, puesto que se produjo dos días antes de la celebración de elecciones generales. Los ganadores de las licencias en la banda de los 26 GHz fueron 'Broadnet', 'Sky Point', y 'Banda 26', mientras que los de las licencias de 3,5 GHz son 'Firstmark Comunicaciones España', 'Abrared' (antes 'Consorcio Abranet'), y 'Banda Ancha' (antes 'Consorcio Aló 2000'). Estos últimos fueron los que consiguieron tener en su poder las frecuencias en las que posteriormente se podrían ofrecer servicios WiMAX. 'Uni2' y 'Retevisión', no tuvieron que concurrir a concurso, pues les había sido concedido el uso de la banda de frecuencias en torno a 3,5 GHz en 1998. (Fuente: IDG Communications)

En la actualidad, hay cuatro operadores con licencia WiMAX (LMDS en 3,5 GHz) en España, y son:

- 'Clearwire' (antigua 'Banda Ancha')
- 'Neo Sky' (antigua 'Abrared' y 'Sky Point', propiedad de Iberdrola)
- 'Iberbanda' (antiguo 'Firstmark', propiedad de Telefónica)

- ‘ONO’ (parte de la antigua ‘Retevisión’, operador de cable propiedad varios fondos de inversión de distintas nacionalidades). ‘Euskaltel’ emplea la licencia de ‘ONO’ en el País Vasco.

En la página web del WiMAX *Forum* ([www.wimaxforum.org](http://www.wimaxforum.org)) se puede acceder a un mapa mundial en el que aparecen localizados todos los operadores que ofrecen acceso WiMAX. Si nos fijamos en España, aparecen tres operadores disponibles, todos ellos operando en la banda de 3,5 GHz:

1. Uno de los operadores que aparece es ‘Iberbanda’ que se refleja dos veces en el mapa por ofrecer dos tipos de servicios:
  - los regidos por el estándar 802.16d, con la compañía ‘Alvarion’ como proveedor de dispositivos.
  - los regidos por el estándar 802.16e, con la compañía ‘Alcatel-Lucent’ como proveedor de dispositivos.
2. Otro de los operadores que aparece es ‘Euskaltel’, que ofrece servicios regidos por el estándar 802.16d.
3. El último de los operadores que aparece en el mapa es ‘Clearwire’, que ofrece servicios regidos por el estándar 802.16e con la compañía ‘Alvarion’ como proveedor de dispositivos.

Además de los citados operadores, hay muchas compañías que, a través de sus páginas web, aseguran ofrecer WiMAX, como por ejemplo: ‘portalWiMAX’, ‘OSF Xarxa de Telecomunicacions’, ‘Esystem’, ‘Embou’, ‘Megavista’, etc. Sin embargo, durante la realización del proyecto, tratamos de contactar con alguna de éstas para que nos confirmaran el tipo de tecnología que ofrecían. Lo único que se pudo sacar en claro es que, al menos, las compañías ‘Megavista’ y ‘OSF Xarxa de Telecomunicacions’ lo que ofrecían realmente era lo que ellos denominan ‘PreWiMAX’, que será probablemente LMDS.

En este sentido hay que tener especial cuidado, pues hay muchas compañías que se están aprovechando del desconocimiento total por parte de la población de las verdaderas características que debe tener un acceso WiMAX. Algunas de estas compañías dicen ofrecer acceso WiMAX cuando en realidad lo que ofrecen es acceso a Internet inalámbrico a través de LMDS o incluso de 802.11n (Wi-Fi mejorado). Se recomienda consultar un interesante artículo llamado “*Pre-WiMAX o Post-WiFi: ¿nos están engañando?*” que aparece en el blog (<http://albentia.wordpress.com>) de la compañía ‘Albentia Systems’, fabricante de equipos WiMAX. Es el citado artículo nos dan algunas claves con las que podemos identificar cuando se nos está ofreciendo realmente tecnología WiMAX y cuando no.

### **3.5.1. OPERADORES DE WiMAX EN ESPAÑA.**

#### **Euskaltel**

Como ya mencionamos, el operador ‘Euskaltel’ emplea la licencia WiMAX de ONO para ofrecer esta tecnología. La tecnología WiMAX de ‘Euskaltel’ acerca la banda ancha a municipios donde hasta ahora, debido a su ubicación, no era posible acceder a

este tipo de servicios. Esta compañía ofrece banda ancha WiMAX con tres opciones de velocidad (300 Kbps/150 Kbps, 1Mbps/600 Kbps ó 2Mbps/600 Kbps) en más de 160 municipios del País Vasco. (Más información: <http://www.euskaltel.com/>)

### **Clearwire**

‘Clearwire’ opera en la ciudad de Sevilla y alrededores. Esta compañía ofrece su producto ‘Instanet’ cuyo lema es: “lo pides, lo enchufas y listo”. En su página web, nos explican que una vez tengamos el módem, lo debemos enchufar a la red eléctrica y al ordenador. Y listo para navegar. No necesita configuración de ningún tipo de software ni un técnico que le instale el servicio. Nos aseguran que con ‘Instanet’, podemos conectarnos a Internet desde cualquier lugar que esté dentro de cobertura ‘Clearwire’. También se puede utilizar el servicio desde distintos ordenadores dentro de nuestra casa; el módem es compatible con cualquier *router* y puede crearse una red local propia. Ofrecen por el momento una conexión de 1Mbps. (Más información: <http://www.clearwire.es/>)

### **Iberbanda**

Para el acceso a Internet a través del WiMAX de Iberbanda, es necesaria la instalación de un pequeño panel en el exterior del edificio. Este panel exterior estará conectado mediante un cable al equipo de interior en el que se conectarán los equipos informáticos. ‘Iberbanda’ ofrece servicio en todas las comunidades autónomas excepto Islas Baleares, Islas Canarias, Ceuta y Melilla. Ofrece productos que permite conexiones desde 1 Mbps a 4Mbps. (Más información: <http://www.iberbanda.es/>)

### **Neo Sky**

Como hemos podido comprobar, esta compañía ni siquiera aparece en el mapa del WiMAX forum que hemos consultado al inicio del apartado 4.6. Esto se debe a que Neo Sky no se ha animado por el momento a ofrecer WiMAX comercialmente. Desde su página web ofrecen como producto LMDS; aunque parece que no han abandonado del todo WiMAX pues, al menos hasta septiembre de 2008, estuvieron realizando pruebas en Madrid con esta tecnología. (Más información: <http://www.neo-sky.com/>)

### 3.6. PROYECTOS Y CASOS PRÁCTICOS WiMAX.

#### 3.6.1. PROYECTOS Y CASOS PRÁCTICOS EN ESPAÑA.

En la actualidad, prácticamente en la totalidad de las comunidades autónomas de España se está desarrollando algún proyecto basado en WiMAX. Sin embargo, la mayoría de estos proyectos se entran en prestar servicios de banda ancha en entornos corporativos y municipales. Todavía se debe evolucionar hacia servicios de conectividad móvil, y servicios personales de banda ancha. A continuación se mencionan algunos de los proyectos que se están llevando a cabo a través de WiMAX en España.

##### VALENCIA

**1. UNIVERSIDAD:** El instituto ITACA de la Universidad Politécnica de Valencia es pionero en probar la tecnología WiMAX en esa comunidad. Tras la instalación de una antena de emisión en septiembre de 2005, se hicieron las primeras mediciones, alcanzando una cobertura de 20 kilómetros a la redonda con unas velocidades medias de 10 Mbps. El 18 de octubre de 2005, ITACA presentó esta tecnología y realizó una demostración en vivo durante las I Jornada de Servicios de Movilidad. En la página web de ITACA (<http://www.itaca.upv.es/movilidad/mainmenu.html>) podemos ver un video de las conferencias.

**2. SAGUNTO:** La Comunidad Valenciana premió en Mayo de 2009, el proyecto “Sagunto WiMAX”, del Ayuntamiento de Sagunto, por facilitar el acceso de los ciudadanos a la Sociedad de la Información. Llevado a cabo en colaboración con ‘Alvarion’, compañía desarrolladora y suministradora de soluciones WiMAX, el proyecto consistió en la implementación de una red municipal de banda ancha para servicios WiMAX que ha posibilitado la creación de un sistema de control de tráfico y de seguridad en edificios públicos. De ello se encargan 73 cámaras, de las que 64 son exteriores (ubicadas en las vías públicas) y 9 son interiores (en los edificios municipales), que permiten controlar en tiempo real 15 rotondas, 60 intersecciones o cruces y 40 tramos de distintas avenidas de la ciudad.

La implementación de la red de servicios radio WiMAX fue preferida en relación con otras tecnologías por su mayor ancho de banda, más rentabilidad, mayor alcance, más calidad de servicio y más seguridad.

(Más información: <http://www.ticpymes.es/Noticias/General/200906080013/Sagunto-se-convierte-en-el-mejor-municipio-impulsor-de-las-TIC.aspx>)

##### EXTREMADURA

Una red WiMAX se ha implantando como proyecto piloto, instalándose en los campus universitarios de Cáceres y Badajoz. Esta red es la que posibilita el acceso a la red de datos de la Universidad desde los exteriores del campus. Se ha llevado a cabo la instalación de dos antenas, una en el Centro de Cáceres y otra en el Centro de Badajoz para dar una cierta cobertura desde las ciudades a la red de datos de la Universidad de Extremadura. La red se encuentra operativa desde el día 30 de septiembre de 2005 en el Rectorado, la Facultad de Ciencias Económicas y Empresariales y la Biblioteca Central

del campus de Badajoz. Después se han incorporado el resto de centros de la Universidad.

(Más información:

<http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=349>)

## **BALEARES**

Telefónica España pone en funcionamiento la primera experiencia mundial de navegación simultánea en el mar y por Internet: la estación base WiMAX ha sido ubicada en un emplazamiento en la Sierra de Na Burguesa, en Palma de Mallorca, y desde ella se proporciona la cobertura a toda la bahía de Palma.

Para realizar la prueba piloto de este servicio, un equipo de cliente WiMAX fue instalado en el Velero Rafael Verdura, de 23 metros de eslora, en el que se ha situado la antena en la parte superior del palo de mesana. En todo momento, se ha conseguido navegar por Internet vía WiMAX a velocidades medias de 2 Mbit/s.

(Más información:

<http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=349>)

## **ANDALUCÍA**

**1. GATA (ALMERÍA):** El Ayuntamiento de Gata (Almería) ha aprobado la creación de una red de cámaras de vigilancia conectadas mediante WiMAX, con el fin de prevenir el vandalismo. Además, se aprovechará la infraestructura para proporcionar acceso a Internet inalámbrico a los vecinos por precios de entre 7 y 12 euros.

(Más información: <http://blogwimax.com/2009/03/01/el-ayuntamiento-de-gata-instalara-camaras-conectadas-por-wimax-para-prevenir-el-vandalismo/>)

**2. NÍJAR (ALMERÍA):** En Almería, la empresa 'Ibersystems' ha diseñado e implementado una red WiMAX de cobertura extensa para control de regadío en Almería.

La comunidad de regantes de Níjar ha modernizado su red de suministro de agua gracias a un sistema de control de regadío que comunica las válvulas de la nueva red de tuberías con un centro de control. Se ha creado una red WiMAX que cubre una extensión de unos 350 km<sup>2</sup>. La red dispone de un enlace WiMAX de gran velocidad entre los tres nodos principales de cobertura. Desde esos nodos se da cobertura a los 38 puntos clave a unir. Estos puntos remotos están equipados con válvulas que abren o cierran pasos entre tuberías, cámaras de seguridad que vigilan el estado de las balsas o sensores de presión.

La red WiMAX une 38 puntos con el centro de control, situado en la Estación Concentradora número 29. Ese punto central está equipado con una torreta de 20 metros de altura y dispone de 3 estaciones base *Alvarion BreezeAccess VL* dando cobertura a su alrededor. Siempre se evita sobrepasar una distancia de más de 5 km entre una estación base y una estación remota y siempre se dispone de visión directa entre puntos. Con esto se consigue trabajar en frecuencias de banda libre respetando los límites máximos de potencia marcados por la ley.



Cada punto de la red dispone de 6 Mbps de ancho de banda y el enlace WiMAX troncal soporta hasta 70 Mbps de velocidad, suficiente para visualizar las cámaras IP instaladas en las balsas de las Estaciones concentradoras y para gestionar todas las válvulas en tiempo real.

Gestionar una red de tuberías con esta tecnología permite un despliegue rápido y a bajo coste ya que no se necesitan abrir zanjas para pasar cableado y siempre se pueden aprovechar los armarios de electrónica de las válvulas para colocar la electrónica de los puntos de acceso. Además se trabaja con tecnologías estándar y no tecnologías propietarias de comunicación. Algunas ventajas que proporciona esta red WiMAX son:

- Migración de sistemas SCADA con protocolos propietarios a una infraestructura estándar totalmente ampliable.
- Las obras son mínimas en comparación con las opciones cableadas.
- Implantación muy rápida.
- El coste de mantenimiento es mínimo ya que se trabaja con sistemas estándar.
- Se puede ampliar la red de forma más rápida.
- Seguridad contra accesos no deseados
- Infraestructura común para diferentes servicios (VozIP, Internet, Videovigilancia)

(Más información:

[http://www.ibersystems.es/cobertura\\_red\\_wimax\\_nijar\\_control\\_de\\_regadio.aspx#](http://www.ibersystems.es/cobertura_red_wimax_nijar_control_de_regadio.aspx#))

**3. MIJAS (MÁLAGA):** El pueblo malagueño de Mijas es en realidad tres pueblos, uno en la sierra y dos en la costa, separados entre sí. Para dar acceso a Internet a sus 22 oficinas municipales, el ayuntamiento mijeño tenía dos opciones: o se gastaba una fortuna en 22 líneas ADSL o implantaba WiMAX. El plan, enmarcado en la iniciativa “Mijas Digital”, pretende unir entre sí las dependencias que el ayuntamiento tiene dispersas por el municipio.

(Más información: <http://www.mijas.es/mijasdigital/pagina.asp?pag=218>)

**4. CÁDIZ:** La Universidad de Cádiz inició en el mes de agosto de 2007 la instalación de las antenas que permitirían distribuir la nueva red inalámbrica, basada en la tecnología WIMAX, en los entornos de los campus de la UCA.

El propósito de esta red es permitir llevar la red de datos de la Universidad de Cádiz a todos los usuarios y miembros de la comunidad universitaria que se encuentren situados en un perímetro de varios kilómetros de las antenas.

Para utilizar la nueva red inalámbrica externa se ha puesto a disposición de los usuarios dos tipos de receptores. En primer lugar, un receptor de altas prestaciones, pensado para la organización de eventos universitarios en recintos ajenos a la UCA, que convierte la señal WiMAX en una red Wi-Fi convencional de 54 Megas multiusuario. El otro tipo de dispositivo se destina al uso individual de los posibles usuarios y permite acceder a la red, en cualquier punto de cobertura, con un ancho de banda de 6 megas.

(Más información: <http://blogwimax.com/2007/08/07/la-universidad-de-cadiz-instala-una-red-wimax/>)

**5. OTRAS LOCALIDADES:** En 2005, ‘Iberbanda’ comenzó en Andalucía la primera experiencia comercial de WiMAX con tecnología de ‘Intel’ en Europa. Para ello, la compañía instaló equipos de ‘Alvarion’ con el chip ‘Rosedale’ en medio centenar de clientes de varias localidades de Almería: Canjajar, Instinción, Ohanes, Padules y Ragol.

(Más información:

<http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=349>)

## MADRID

**1. CIUDAD:** La CMT (Comisión del Mercado de las Telecomunicaciones) ha autorizado bajo ciertas condiciones que la EMT (Empresa Municipal de Transportes) ofrezca Internet en su red de autobuses públicos. El Ayuntamiento de Madrid comenzará a implantar un sistema de acceso inalámbrico a Internet en sus autobuses que se suponía iba a estar listo para antes del verano de 2009.

La señal llegará a cada autobús por ondas de radio a través de tecnología WiMAX y después se repartirá dentro de cada uno de ellos mediante Wi-Fi. La CMT ha establecido una serie de condiciones, para el desarrollo de este proyecto, dirigidas a respetar el libre mercado y que consistirían en, o bien, crear un servicio financiado a través del cobro un suplemento en el billete de autobús a todos aquellos viajeros interesados en navegar por Internet durante el trayecto, o bien, ofrecer un servicio de Wi-Fi gratuito subvencionado mediante un sistema publicitario. El Ayuntamiento de Madrid, de resultar exitoso este proyecto, podría extender el futuro servicio de acceso inalámbrico a Internet en autobuses también al metro de la ciudad.

(Más información: <http://www.desarrolloweb.com/actualidad/wifi-publica-autobuses-madrid-1521.html>)

**2. ALCORCÓN:** La empresa 'Neomedia' ha realizado en Alcorcón (Madrid), un proyecto orientado a la implementación de una red WiMAX para el ayuntamiento de la citada localidad. Mediante este proyecto, el Ayuntamiento de Alcorcón ha puesto en marcha un sistema de video-vigilancia en edificios municipales y colegios mediante WiMAX. La red está formada por más de 70 radio-enlaces, que se han llevado a cabo a través de antenas de 'Alvarion', que dan soporte a más de 100 cámaras. Cuando se detecta movimiento, el sistema se encarga de avisar a la policía.

(Más información:

[http://www.clubdeinnovacion.es/index.php?option=com\\_mtree&task=viewlink&link\\_id=76&Itemid=80](http://www.clubdeinnovacion.es/index.php?option=com_mtree&task=viewlink&link_id=76&Itemid=80))

## CATALUÑA

La empresa 'Neomedia' ha desplegado un *backhaul* WiMAX que da soporte a la red inalámbrica mallada del Ayuntamiento de Barcelona.

Por otro lado, la ciudad de Vic (Barcelona) dispone de un sistema que permite vigilar los puntos más conflictivos de la urbe y sus principales vías de entrada y salida, gracias a un proyecto llevado a cabo por 'bcSistemas' con productos de 'Alvarion'. Dicha solución consta de una red inalámbrica de alta capacidad que permite la conectividad de servicios WiMAX para vídeo IP en tiempo real y en todo el municipio, basándose en sistemas punto a punto *BreezeNET B* y punto-multipunto *BreezeACCESS VL* de 'Alvarion'. La solución, que integra el circuito de televisión analógico que ya existía, posibilita la interconexión de cámaras para video-vigilancia del tráfico urbano, y está totalmente controlada y gestionada por los agentes de la Guardia Urbana de Vic, obteniendo así control y rápida capacidad de reacción ante incidencias, así como una mejor gestión de los recursos.

(Más información: <http://www.redestelecom.es/Noticias/200901270018/La-Guardia-Urbana-de-Vic-implanta-una-red-de-vigilancia-con-las-soluciones-de-Alvarion.aspx>)

### **CASTILLA LA MANCHA:**

**1. LA MANCOMUNIDAD LA MANCHA:** integrada por Alhambra, Carrizosa, Llanos, La Solana, Manzanares, Membrilla, Ruidera, San Carlos del Valle y Villarta de San Juan, y perteneciente a la provincia de Ciudad Real, está desarrollando actualmente un proyecto de infraestructura de comunicaciones que le permitirá mejorar los servicios prestados a los ciudadanos. Los municipios de la zona estarán unidos a través de una infraestructura de Banda Ancha inalámbrica, cuyo punto central se ubicará en la localidad de La Solana, desde donde se comunicará con el resto de emplazamientos. 'Nostracom', la compañía encargada de llevar a cabo el proyecto, utiliza la tecnología WiMAX en su versión 802.16d.

(Más información: <http://www.noticiascadadia.com/noticia/11759-los-municipios-de-la-mancomunidad-la-mancha-estaran-intercomunicados-a-traves-de-una-infraes/>)

**2. VALDEPEÑAS:** La localidad ciudadrealeña de Valdepeñas contará a finales de 2009 con el servicio 'ValdeWi+', una red Wi-Fi y WiMAX municipal a la que se podrá suscribir todo aquel que lo desee con precios públicos a una amplia gama de servicios. Este servicio lo gestionará la empresa 'Cesser'.

El alcalde de la localidad ha explicado que el "monopolio" de las empresas dedicadas a prestar el servicio de banda ancha ha provocado su encarecimiento, por lo que el servicio que se presenta pretende rebajar el precio habitual hasta en un 300 por cien, ya que el coste para el cliente no superará los 25 euros en el mejor de los casos.

Una vez que la red se consolide en el casco urbano, la prestación se extenderá a otros barrios con un servicio de banda ancha, y se instalará el primer servicio público de Castilla-La Mancha de alarma para el comercio y para particulares conectado vía WiMAX a un servidor situado en la Policía Local.

La red prestará también el servicio de telefonía fija que inicialmente no se había incluido en el pliego de condiciones para la concesión y cuya tarifa plana se ha fijado en cinco euros mensuales más IVA para las llamadas locales, provinciales y nacionales.

(Más información: <http://blogwimax.com/2009/03/22/valdepenas-pondra-en-marcha-una-red-wimax-wifi/>)

### **3.6.2. OTROS CASOS PRÁCTICOS.**

A continuación se mencionan tres casos prácticos llevados a cabo fuera de España que pueden servirnos para hacernos una idea de la variedad de escenarios en los que tecnología WiMAX puede resultar de gran utilidad.

**HAWAI "IRONMAN":** La WTC (*World Triathlon Corporation*) es el organismo que organiza el campeonato del mundo de triatlón "Ironman" en Hawái. Esta prueba está considerada como la reina de la resistencia en todo el mundo. La WTC quería poder transmitir la prueba desde varios puntos con una gran calidad de imagen. Pero este

organismo, se enfrentaba a un gran número de barreras, y sobretodo a una distancia de 225 kilómetros en terreno difícil y en una localización remota de la isla.

En colaboración con ‘Intel’, ‘Airspan Networks’, ‘Cisco Systems’ y ‘Dell’, la WTC desplegó una red inalámbrica WiMAX a lo largo de todo el recorrido. Se utilizó la tecnología RFID (*Radio-Frequency IDentification*) para seguir el proceso de los participantes a lo largo del recorrido y el alcance y el gran ancho de banda de la red WiMAX para transmitir vídeo de alta calidad (8 Mbps necesarios para una alta calidad) a través de Internet casi en tiempo real. Los espectadores tanto en sus casas como los que se encontraban presentes pudieron ver el vídeo y seguir la progresión de los atletas simplemente conectándose a una página web. También fue posible seguirlo a través de hotspots Wi-Fi con tecnología móvil ‘Intel Centrino’ y a través de PDAs (*Personal Digital Assistants*) con tecnología ‘Intel Xscale’.

El despliegue fue muy dificultoso debido a la orografía de la isla (con equipos incluso en las laderas de los volcanes) ya que hubo que utilizar generadores de señal en zonas donde ésta no podía llegar. Pero el hecho de que resultara un éxito confirmó una vez más las posibilidades de WiMAX en un entorno hostil y no digamos por tanto en entornos favorables. (Fuente: IBM, 2006)

**FESTIVAL DE SUNDANCE:** La empresa ‘Intel’ es uno de los patrocinadores del festival de cine de Sundance y vio una oportunidad de demostrar el potencial de la banda ancha inalámbrica para cambiar la manera en que la industria cinematográfica distribuye sus contenidos a las salas

‘Intel’, en colaboración con ‘Alvarion’ y ‘Mountain Wireless’, diseñó un conjunto de experiencias para los visitantes del festival del año 2005, todas ellas basadas en tecnología WiMAX. La red incluía un enlace *backhaul* inalámbrico punto a punto de 88 kilómetros desde Salt Lake City hasta Park City, en el estado de Utah (Estados Unidos). La distribución dentro de la sala de Park City se realizó mediante una red inalámbrica de banda ancha punto-multipunto. Se trataba de la primera vez que se presentaba una película a través de una conexión inalámbrica a Internet a un público que se encontraba en una localización remota. Junto con hardware disponible en el mercado y una instalación sencilla, la calidad de imagen resultante no se distinguía de la imagen observada si esta hubiera sido emitida en la propia sala. (Fuente: IBM, 2006)

**AGUJA DE SEATTLE:** La empresa ‘Speakeasy’ fue la encargada de realizar en 2005 lo que hasta el momento era el mayor despliegue de tecnología WiMAX. Su ubicación es la conocida aguja del *skyline* de Seattle (Washington, EEUU) y los edificios adyacentes, ofreciendo así cobertura al centro de negocios de esta ciudad. Era la primera vez que se establecía un servicio de banda ancha inalámbrico en un entorno de gran densidad y con un servicio *broadcast* punto-multipunto. (Fuente: IBM, 2006)

### 3.7. CONCLUSIONES.

Al contrario de lo que se pudiera pensar, y como hemos visto, no está siendo el ámbito corporativo el primero en probar la tecnología WiMAX. Los gobiernos autonómicos ven en la nueva tecnología una oportunidad buena y barata para poder dar acceso a Internet a muchos de sus municipios como un servicio público más.

Durante años, el WiMAX Forum ha puesto todo su empeño para que el estándar 802.16 se viese como una muy buena opción de futuro para acceder a través de banda ancha de modo inalámbrico a Internet. En el mercado “fijo” pocos se atreven a discutir las enormes ventajas de WiMAX: antenas con gran alcance, velocidades que en la versión 802.16-2009 llegan hasta los 300 Mbps teóricos, mejores mecanismos de seguridad que Wi-Fi, etc. Sin embargo, el mercado de las telecomunicaciones parece distinguir cada vez menos entre tecnologías de acceso fijo, o móvil. Los usuarios buscan ahora una única solución que les permita acceder inalámbricamente a Internet tanto desde sus hogares como desde cualquier punto de su ciudad. Es en el sector de las comunicaciones móviles donde a WiMAX le surgen mayores competidores, poniéndolo en una situación que provoca que incluso muchos duden de su futuro.

Muchas corporaciones han invertido grandes cantidades de tiempo y dinero en su apuesta por el desarrollo de WiMAX, y se encargan ahora de defender a capa y espada las ventajas que esta tecnología ofrece sobre las actuales redes móviles de tercera generación. Recuerdan que WiMAX es tres veces más potente que estas tecnologías y además cuesta la mitad.

No hay que saber mucho del negocio de las telecomunicaciones como para percatarse de que en España, en la actualidad, se pagan precios abusivos por el acceso al servicio de Internet móvil 3G que ofrecen los operadores de telefonía. Y es que, desde el despliegue de la tercera generación, han pasado casi diez años y estos todavía no han recuperado su inversión. Por ello los defensores de WiMAX, aprovechan esta situación para recalcar la importancia que tiene el precio. Subrayan que el deseo de los usuarios es poder disponer de un servicio de acceso a Internet, que permita también realizar llamadas, me sirva en casa o donde sea y a un precio asequible.

Aquellos que han apostado por WiMAX, alimentan su esperanza de éxito poniendo la vista en países como Corea del Sur. En este país, que se considera el país del mundo más avanzado en telefonía móvil, ya había más de 100.000 abonados al WiMAX móvil a principios de 2008, pagando una tarifa plana de 21 euros al mes (según los datos que aportó Ron Resnick, presidente del WiMAX Forum, en la entrevista que concedió al periódico El País en Febrero de ese año). En esa misma fecha, había más de 40 productos que incorporaban WiMAX; como módems USB, *routers* o móviles.

Pero fabricantes de redes y terminales, y operadoras ya están practicando pruebas piloto para comprobar la eficacia de algunas de las tecnologías que pugnan por liderar la cuarta generación. Y es que bajo ese nombre, se enfrentan principalmente dos tecnologías: LTE y WiMAX. Ambos contendientes coinciden en garantizar velocidades comprobables de más de 100 Mbps.

La tecnología LTE (*Long Term Evolution*) se propone como una mejora de la tercera generación de móviles (y también, por supuesto, de la conocida como 3,5G). Tiene detrás a empresas como ‘Nokia Siemens Networks’ realizando pruebas de campo en

zonas urbanas a 173 Mbps. Se prevé que esté disponible a partir de 2011. En España operadoras como 'Telefónica' o 'Vodafone' ya está haciendo pruebas con la tecnología LTE aunque por el momento no hablan de fechas relativas a ofertas comerciales. (Fuente: Actualidad Informática).

En el caso de WiMax, en su versión IEEE 802.16-2009, las implementaciones pueden ser usadas tanto para servicios fijos como móviles. La telefónica 'Sprint Nextel' en Estados Unidos invirtió 5 mil millones de dólares en desplegar su red WiMax y hay muchas corporaciones, como 'Vodafone', 'Intel' o 'Telmex', que han puesto sus fichas en ella (Fuente: Actualidad Informática).

El argumento que más esgrimen los impulsores de WiMAX se basa en que la principal ventaja de esta tecnología frente a LTE, es que WiMAX es ya una realidad, ya se ha puesto en marcha; con lo que lo más difícil de superar, ya se ha superado. Curioso es que defiendan WiMAX frente a LTE sin alegar cuestión tecnológica alguna, simplemente vienen a decir que WiMAX es mejor porque llegó antes. En realidad no es de extrañar, porque según afirman los conocedores del nuevo estándar LTE, éste y WiMAX son dos tecnologías que apenas se diferencian.

'Intel' es, sin duda, una de las empresas referentes cuando se trata de WiMAX y de su evolución en el mercado, por eso es uno de los agentes que puede actuar para que WiMAX realmente compita con LTE y no caiga en el olvido. La tensa situación que se vive provoca que cualquier declaración de los ejecutivos de 'Intel' se tome como una señal directa o indirecta de hacia donde se dirige esta tecnología. Recientemente, Sean Maloney, director de ventas y marketing de 'Intel', declaró en junio de 2008 a la BBC (*British Broadcasting Corporation*) que en vez de pelearse, lo natural sería que LTE y WiMAX se fusionaran en una sola tecnología. Los más extremistas han visto reflejado en este comentario la poca confianza que tiene 'Intel' en el futuro de WiMAX.

Para colmo, 'Nokia', el principal fabricante de teléfonos del mundo y casualmente miembro fundador del WiMAX Forum, dejó hace ya casi un año de producir el teléfono móvil "*N810 WiMAX Edition*". La compañía finlandesa había lanzado este aparato hacía tan sólo nueve meses. 'Nokia' no descartó lanzar más teléfonos con WiMAX en el futuro: "Continuaremos siguiendo la tecnología y su evolución", añadió el portavoz de la marca. (Fuente: Informador.com)

Todos estos hechos están transmitiendo una sensación de fracaso de WiMAX incluso a los usuarios que esperaban impacientemente su llegada y todavía ni siquiera han podido probar una tecnología que ya muchos dan por muerta. Es una muestra más del vertiginoso ritmo de "actualización" al que hemos llegado en el sector de las telecomunicaciones, en el que la experiencia no es siempre un grado.

A pesar de que algunas compañías ya están trabajando en iniciativas para la implantación de LTE, los analistas apuntan a que habrá que esperar unos cinco años para encontrarse con el gran éxito de LTE en el mercado. En la batalla dialéctica WiMAX siempre tiene las de perder cuando se enfrenta a LTE. El volumen de esta última tecnología será de tal magnitud que da igual que WiMAX ofrezca mejores prestaciones tecnológicas o que sus costos de propiedad intelectual sean menores, o que ya haya operadores comercializando ya la tecnología y LTE aún se mueva por el laboratorio. Por tanto, a WiMAX tan sólo le queda la opción de seguir mejorando sus estándares, tal y como han hecho hasta ahora, y aprovechar esa ventaja temporal, que tanto mencionan sus defensores, para ofrecer y publicitar de verdad servicios y soluciones que conquisten al mercado de las comunicaciones móviles de banda ancha.



## **CAPÍTULO IV**

# **DESARROLLO DE UN SOFTWARE DE VIDEOCONFERENCIA ESPECÍFICO**

Comenzaremos la parte práctica del proyecto centrándonos en el programa informático que nos permitirá realizar la videoconferencia. Conviene recordar brevemente el escenario concreto que planteábamos al inicio de este documento.

El escenario se compone de dos ubicaciones. En la primera de ellas, un aula de un centro de estudios, se encuentran un grupo de alumnos acompañados de un profesor. En la segunda ubicación estará situado únicamente un profesor. Esta última ubicación no es un lugar concreto sino que de lo que se trata es de que pueda ser prácticamente cualquier lugar. Ambos extremos deberán contactar entre sí enviándose audio y vídeo a través de Internet.

En el primer apartado de este capítulo pasaremos a recordar con más detalle el escenario que acabamos de mencionar. De este modo nos será más fácil definir los requisitos necesarios del programa de videoconferencia que debemos usar para dar solución a los problemas planteados.

Finalmente se presentará el software propio de videoconferencia, “Teletrófono”, desarrollado íntegramente en el lenguaje de programación Java. Comentaremos su funcionalidad y se explicarán brevemente los conceptos fundamentales de JMF (*Java Media Framework*), la API (*Application Programming Interface*) de Java usada para desarrollar el programa. Por último nos centraremos en explicar, paso a paso, cómo se ha llevado a cabo tanto el diseño como la implementación de Teletrófono.



## **4.1. REQUISITOS DEL SOFTWARE DE VIDEOCONFERENCIA.**

En base a la situación planteada por el proyecto, el programa que usemos para llevar a cabo la videoconferencia deberá cumplir una serie de requisitos indispensables que a continuación vamos a recordar.

El propósito de la situación planteada por el proyecto es que el profesor situado en el extremo remoto sea capaz de transmitir la máxima información posible sobre el entorno en el que se encuentra, por tanto, será indispensable que este lado pueda transmitir información tanto auditiva como visual al otro extremo. El profesor y alumnos en el aula, en el extremo local, deberán tener asimismo la posibilidad de comunicarse de algún modo con el otro extremo. Dicha comunicación bastará con que sea únicamente de audio. Por tanto, la transmisión del audio deberá ser bidireccional y el vídeo deberá transmitirse, al menos, del lado remoto al local. Nos gustaría también que el profesor en remoto viese en su pantalla de ordenador la imagen que está transmitiendo al otro lado.

Será un aspecto importante que la imagen en el lado local se reciba con un tamaño lo suficientemente grande para que se aprecie bien al profesor en remoto y su entorno. Hay que tener en cuenta que el profesor que está en clase proyectará la pantalla del ordenador para que todos los alumnos vean la imagen que se recibe desde remoto. La imagen recibida deberá ser, por tanto, de un tamaño aceptable.

Sin embargo, tal y como mencionamos en el primer capítulo, buscamos una solución software que nos proporcione más posibilidades aparte de la mera transmisión de audio y vídeo. Es requisito indispensable del programa que nos ofrezca la posibilidad de ir insertando rótulos mientras se desarrolla la videoconferencia. Los rótulos deberían poder insertarse desde el lado local. De esta manera, el profesor presente en el aula podría ir introduciendo letreros que se posicionarían bajo la imagen recibida desde remoto. Otra de las funcionalidades adicionales que pedimos al software de videoconferencia es que nos permita grabar la sesión. Esta grabación deberá incluir tanto el vídeo en el que aparecerá el profesor en remoto llevando a cabo la videoconferencia, como los subtítulos que el profesor en el aula ha ido insertando. De este modo evitamos que la experiencia finalice cuando finalice la videoconferencia. El control de esta grabación también sería deseable que estuviese situado en el lado local para liberar así de carga de trabajo al profesor en remoto.

Necesitamos un software que sea muy sencillo de utilizar. Hemos de tener en cuenta que los usuarios de la aplicación probablemente no sean expertos en informática o telecomunicaciones por lo que será imprescindible una interfaz amigable. De este modo nos evitaríamos el tener que necesitar la ayuda de técnicos especialistas en cada uno de los extremos.

Por último sería deseable que la solución fuese asequible económicamente y fácilmente accesible para cualquier institución. El que la videoconferencia se vaya a realizar a través de una red como Internet nos va a beneficiar en varios aspectos. La elección de esta red configura una solución accesible, pues casi la totalidad de centros de enseñanza en España cuentan ya con acceso a Internet, económica y familiar para la gran mayoría de estudiantes y profesores.

## 4.2. SOFTWARE PROPIO DE VIDEOCONFERENCIA: “TELETRÓFONO”.

“Teletrófono” es el nombre que el ingeniero italiano Antonio Meucci dio en 1870 a su mayor invento. Debido a sus problemas económicos no pudo hacer frente a los gastos necesarios para mantener la patente de éste. En 1876, la empresa telegráfica “Western Union”, a la que Meucci había mostrado años antes su proyecto, presentó el “Teléfono” como la gran invención del escocés Alexander Graham Bell. Como pequeño homenaje a Meucci, “Teletrófono” ha sido el nombre se ha escogido para el programa de videoconferencia que se ha desarrollado. El programa pretende dar solución a los problemas que este proyecto plantea. Tratará de dar cobertura a cada una de las necesidades que surgen en el escenario descrito en anteriores apartados.

El lenguaje de programación escogido para implementar el programa ha sido Java. Java es un lenguaje de programación desarrollado por ‘*Sun Microsystems*’ a principios de los años noventa. Sus principales características son: es un lenguaje orientado a objetos y además independiente de la plataforma.

La primera característica, orientado a objetos, se refiere a un método de programación y al diseño del lenguaje. Los datos y el código, se combinan en entidades llamadas “objetos”. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos). El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar. De este modo se crean entidades más genéricas que permitirían la reutilización de software entre proyectos, una de las premisas fundamentales de la ingeniería del software.

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema, que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente. La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “*bytecode*”. Esta pieza está a medio camino entre el código fuente y el código máquina que entiende el dispositivo destino. El “*bytecode*” es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código.

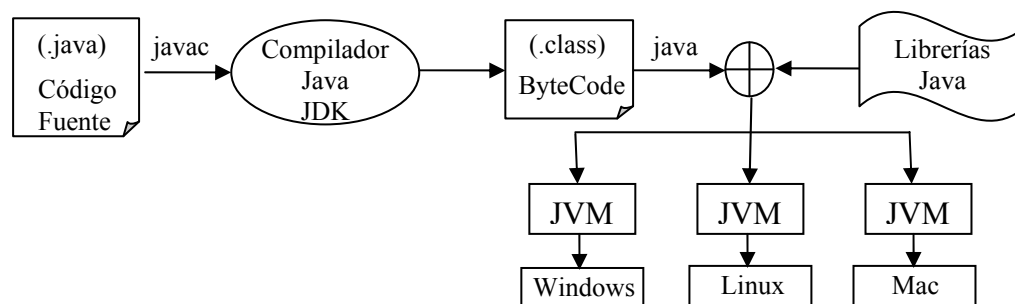


Figura 4.1. Secuencia de ejecución de un código fuente en Java.

Volviendo a nuestra aplicación “Teletrófono”, sabemos que ésta habrá de manejar y trabajar constantemente con fuentes de audio y vídeo. Existe una API (Application Programming Interface) de Java especialmente dedicada a la manipulación y procesamiento de fuentes multimedia. Dicha interfaz se llama *Java Media Framework* (JMF) y va a ser la clave de nuestra aplicación.

Tal y como se comentó al inicio de este documento, cuando se listaban los objetivos del proyecto, uno de los cometidos de este trabajo es evaluar Java y, en concreto, JMF como medio para diseñar e implementar un software de videoconferencia en un contexto educativo. Por tanto, empezaremos concretando las funciones que debe cumplir nuestro programa. La funcionalidad se ha definido de acuerdo a los objetivos marcados para el proyecto y de acuerdo con el resumen que se hacía en el primer apartado de este capítulo acerca de los requisitos que debía cumplir el programa de videoconferencia que se utilizase en el escenario planteado.

Tras la explicación de las funciones que implementa nuestro programa “Teletrófono”, explicaremos el diseño general de éste, los principios básicos que describen la API Java Media Framework y el proceso que hemos seguido para implementar la aplicación, así como los problemas que han ido surgiendo y cómo los hemos solucionado.

Para concluir el apartado entraremos en profundidad en cada uno de los bloques que componen el programa explicando el modo en el que se ha programado el código fuente de éste.

#### **4.2.1. FUNCIONALIDADES.**

“Teletrófono” es un programa dirigido especialmente a docentes de cualquier materia y grado que estén interesados en incorporar la videoconferencia como herramienta educativa habitual. Por este motivo se trata de una aplicación con una interfaz de usuario muy sencilla e intuitiva, que permite a cualquier persona que tenga un nivel informático medio, utilizar cada una de las herramientas que nos proporciona “Teletrófono”.

Si recordamos el escenario que sirve para ilustrar el propósito del proyecto, tenemos un lado local en el que está un profesor con alumnos en un aula. Tenemos también un lado remoto en el que estará un profesor en un lugar arbitrario. Un posible escenario sería el que se muestra en la figura 4.2.

En un principio, vamos a suponer que en el lado local poseen un ordenador, ya sea de sobremesa o portátil, con micrófono y webcam, y un proyector con su pantalla de proyección a través de la cual los alumnos podrán ver en grandes dimensiones la pantalla de dicho ordenador. En el lado remoto supondremos que el profesor dispone de un ordenador portátil con micrófono y webcam. Ambos extremos deberán tener acceso a Internet (ver capítulos 3 y 5).

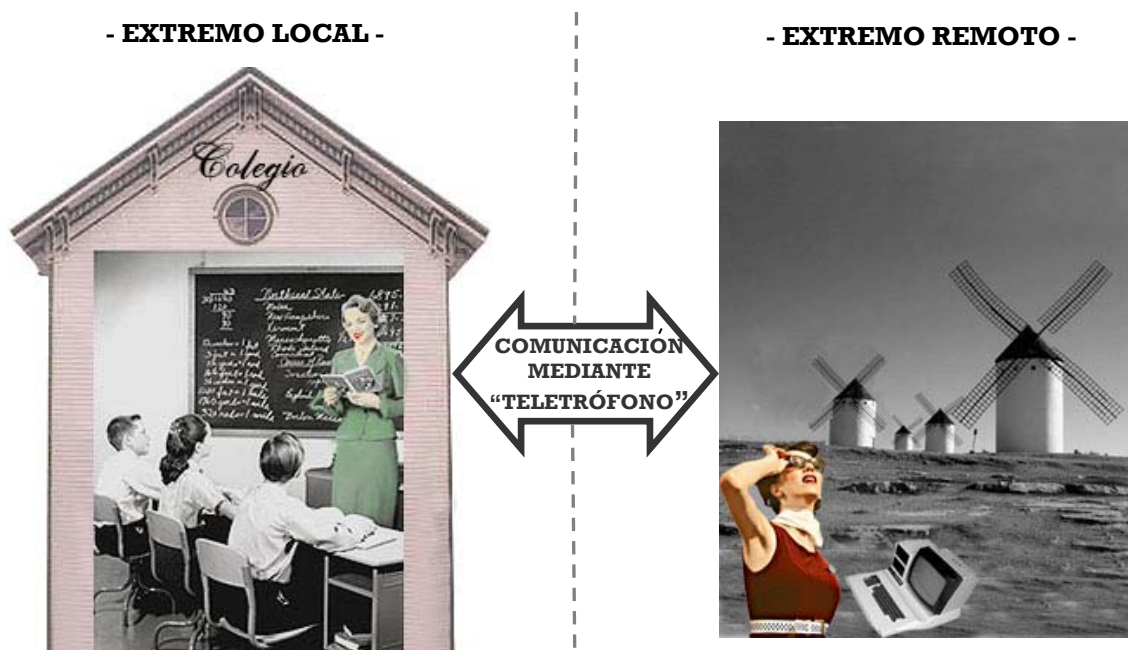


Figura 4.2. Escenario al que se da solución con “Teletrófono”.

La función principal de Teletrófono es la de establecer comunicación audiovisual entre dos lugares separados geográficamente. Sin embargo nos va a proporcionar otra serie de herramientas pensadas especialmente para un escenario docente que darán a la aplicación un valor añadido.

Cuando arrancamos la aplicación nos aparece, tras la pantalla de presentación, la pantalla de “Menú principal” en la que aparecerán las diferentes acciones que se nos permite realizar. Las cuatro acciones permitidas son las siguientes:

1. Iniciar Videoconferencia desde el extremo Local
2. Iniciar Videoconferencia desde el extremo Remoto
3. Reproducir grabación
4. Editar y Re-sincronizar Subtítulos

Para llevar a cabo una videoconferencia, ambos extremos deberán coordinarse, empleando algún mecanismo externo a la aplicación, para lanzar al mismo tiempo desde sus ordenadores nuestro programa Teletrófono.

La primera opción que aparece en el menú principal del programa, “Iniciar Videoconferencia desde el extremo Local”, es la opción que deberá elegir el profesor situado en el extremo local, es decir, el profesor que está en el aula.

La segunda opción del menú, “Iniciar Videoconferencia desde el extremo Remoto”, es la opción que deberá elegir el profesor situado en el extremo remoto, es decir, el profesor que se ha trasladado a un lugar de interés lejos del centro educativo.

Cuando ambos extremos hayan escogido estas opciones y pulsen el botón que inicia la videoconferencia....

- ambos lados podrán mantener una conversación entre sí, pues se iniciará una transmisión de audio bidireccional,
- el profesor y los alumnos presentes en el aula serán capaces de visualizar al profesor que está en el extremo remoto, pues se iniciará una transmisión de vídeo unidireccional desde el extremo remoto al local,
- el profesor en el extremo remoto será capaz de verse a sí mismo, para poder controlar así la imagen que realmente está enviando al otro extremo.

Con estos tres puntos que acabamos de mencionar cumplimos el objetivo más básico del programa: comunicar entre sí los dos extremos.

En primer lugar, la transmisión de audio bidireccional es primordial, siempre indispensable en cualquier escenario de comunicación de este tipo. En segundo lugar, sabemos que el profesor en el aula y sus alumnos serán capaces de ver, a través de la aplicación, al profesor situado en el otro extremo, pudiendo contemplar el lugar al que éste se ha trasladado. El profesor en el lugar remoto podrá transmitir, por tanto, cualquier información tanto auditiva como visual sobre el emplazamiento en el que se encuentra, siendo capaz de comunicar de este modo, una ingente cantidad de información a sus alumnos. Asimismo, como se ha mencionado, el profesor que está retransmitiendo desde un lugar lejano determinado, visualizará en el monitor de su ordenador portátil su propia imagen. Esta característica le será de gran utilidad a dicho profesor para poder comprobar en cada instante qué imagen exacta están recibiendo los alumnos en el aula. Así podrá enfocar a través de la cámara web, en cada momento, la imagen que más le interese de su entorno.

Además, en el extremo local, se dispone de una serie de opciones adicionales:

- el profesor en el aula podrá insertar rótulos bajo la imagen recibida desde remoto para poder ir resumiendo la situación o destacando aspectos a los alumnos presentes en ese momento en el aula,
- el profesor en el aula podrá pulsar el botón de “Iniciar grabación”, comenzando a almacenarse en su ordenador tanto la imagen y el audio recibidos desde remoto como los subtítulos que él mismo vaya introduciendo a partir de ese instante. El vídeo recibido, con la imagen del profesor en el escenario remoto, quedará almacenado en un archivo “.avi” con el nombre escogido y en el directorio deseado. El audio recibido quedará almacenado en el mismo directorio y con el mismo nombre que el vídeo pero con una extensión “.gsm”. Los subtítulos quedarán almacenados en el mismo directorio que el vídeo y con el mismo nombre que éste a excepción de que la extensión del archivo de subtítulos será “.srt”.

El que “Teletrófono” nos proporcione la opción, en el lado local, de insertar rótulos bajo la imagen que se recibe desde remoto, puede dar lugar a muchas posibilidades. El personal docente encargado de realizar la actividad podrá hacer uso de esta herramienta según le convenga.

Una de las posibilidades de los rótulos sería la utilización de éstos a modo imitación de los que aparecen diariamente en los programas televisivos en directo. Normalmente este tipo de rótulos a modo programa de televisión tienen como finalidad resumir la información que el locutor está dando, destacar datos importantes acerca de la noticia, proporcionar datos adicionales no indispensables, informar sobre la ubicación del

locutor, etc. Basándonos en esta inspiración de los directos televisivos podemos, gracias a los rótulos que proporciona “Teletrófono”, poner en práctica este estilo en una actividad educativa propia.

Otra de las posibilidades que los letreros que nos permite introducir “Teletrófono” nos proporciona sería usarlos como puros subtítulos. Es decir, orientados hacia aquellos alumnos que presentan déficit auditivo. En este caso el cometido de los subtítulos sería simplemente transcribir las palabras que el locutor va articulando.

Pero las posibilidades que esta herramienta nos proporciona son infinitas. Como último ejemplo o inspiración para su posible uso, sería la de su utilización como excusa para dar lugar a otra actividad. Por ejemplo, podríamos ir introduciendo en directo, como subtítulos, preguntas acerca de lo que nos está contando el profesor en remoto. Por ejemplo: “¿En qué año se construyó la catedral en la que se encuentra el profesor Gómez?”. Los alumnos deberían ir contestando en un folio a dichas preguntas, recogiendo cada ejercicio al finalizar la sesión. Sería una manera de asegurarse la atención de los alumnos. Pero como ya hemos comentado, el aprovechamiento de esta herramienta en mayor o menor modo depende principalmente de la imaginación del profesor.

Por otro lado la opción que “Teletrófono” nos proporciona para poder grabar tanto la imagen y la voz del profesor en remoto, como los subtítulos que desde el aula alguien fue introduciendo, abre la puerta a multitud de posibilidades educativas. La grabación es una herramienta muy importante principalmente porque nos permite aprovechar la experiencia aún cuando ésta ya ha finalizado. Del mismo modo que ocurre con los rótulos, la imaginación del profesor es clave a la hora de aprovechar en mayor o menor modo esta opción que el programa nos brinda.

Cualquiera de los extremos podrá finalizar la videoconferencia cuando desee pulsando el botón “Finalizar Videoconferencia”.

Un esquema muy sencillo sobre el tipo de comunicación que se lleva a cabo entre los dos extremos con “Teletrófono” sería el siguiente:

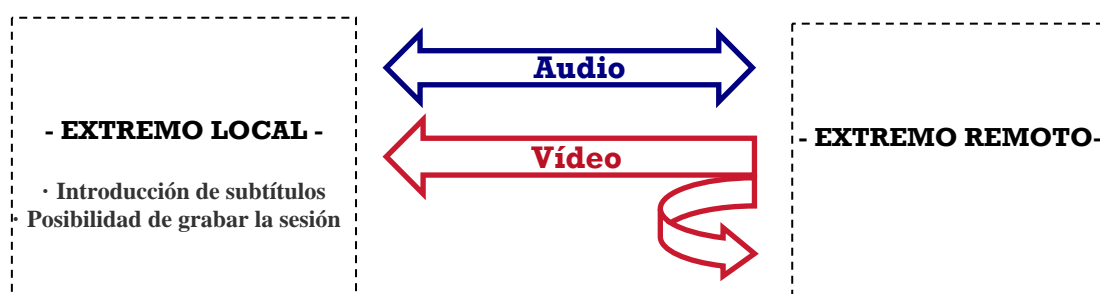


Figura 4.3. Esquema básico de videoconferencia a través de “Teletrófono”.

Pasamos a la tercera de las opciones que nos ofrece el menú principal de “Teletrófono”, “Reproducir grabación”. Esta opción está especialmente pensada para reproducir las grabaciones de las sesiones de videoconferencia, aunque reproducirá también cualquier archivo de vídeo con la extensión ‘.mov’, ‘.avi’ o ‘.mpg’. El reproductor nos preguntará qué archivo de vídeo queremos reproducir y, una vez que lo hayamos escogido, buscará

si dicho archivo tiene un archivo de audio ‘.gsm’ asociado o un archivo ‘.srt’ de subtítulos asociados. En caso de que no haya ningún archivo de audio asociado reproducirá únicamente el archivo de vídeo escogido (con sus subtítulos si es que tiene un archivo de este tipo asociado). En caso de que encuentre un archivo de audio ‘.gsm’ asociado reproducirá simultáneamente tanto el vídeo como el audio (con sus subtítulos si es que tiene un archivo de este tipo asociado).

La última de las opciones que nos ofrece “Teletrófono” es “Editar y Re-sincronizar Subtítulos”. Este bloque de la aplicación nos pedirá, al igual que el reproductor, un archivo de vídeo a editar. El editor de subtítulos sólo trabajará con grabaciones que hayan sido creadas con “Teletrófono”, por tanto, sólo trabajará con archivos de vídeo ‘.avi’, cuyas dimensiones máximas sean 352x288 píxeles y que tengan asociado un archivo de audio ‘.gsm’. Además se comprobará si el archivo de vídeo escogido tiene subtítulos asociados, si no es así, no se podrá iniciar el editor.

Una vez hayamos proporcionado al editor un archivo de vídeo válido nos aparecerá la pantalla de edición. En dicha pantalla se mostrará un reproductor con el que podremos reproducir el vídeo escogido y sus subtítulos. También aparecerá una lista con todos los subtítulos asociados a la grabación y los tiempos de inicio y fin de cada uno de ellos. Para modificar el texto de cualquiera de los subtítulos o su tiempo de inicio o de fin, bastará con seleccionar el subtítulo en la lista y pulsar el botón “Editar”. Nos aparecerá entonces un panel de edición donde podremos realizar los cambios pertinentes.

La herramienta de edición nos va a permitir, por tanto, modificar aquellos subtítulos que se insertaron durante la sesión de videoconferencia. Se nos permite moverlos en el tiempo o modificar su contenido. Esta opción que nos ofrece “Teletrófono” puede sernos muy útil en el caso de que durante la sesión de videoconferencia hubiésemos cometido alguna errata en el texto de algún rótulo. Pero también puede sernos de utilidad simplemente cuando queramos que los subtítulos que figuren junto a la grabación de la sesión, sean totalmente distintos a los que se insertaron en directo, bien por capricho o bien porque a partir de esta opción pueden surgir de nuevo otras formas de actividades educativas.

Estas son, por tanto, las funcionalidades que nos ofrece “Teletrófono”. Como se habrá podido observar, la imaginación del profesor será un elemento clave en el uso de esta aplicación de videoconferencia. Aunque el programa no proporciona un elevado número de herramientas, permite, con unas pocas, realizar multitud de actividades orientadas a la docencia. Se pretende, de este modo, introducir las nuevas tecnologías y, en concreto, la videoconferencia, como medio educativo habitual en los centros de enseñanza de cualquier grado y especialidad.

#### **4.2.2. DISEÑO.**

El diseño de “Teletrófono” se ha desarrollado desde un principio teniendo muy en cuenta la clara división del escenario general en dos “sub-espacios” a los que hemos llamado desde el comienzo “extremo local” y “extremo remoto”. El extremo local será el aula en la que se encuentran los alumnos acompañados de un profesor y el extremo remoto será un lugar de interés en el que se encuentra otro profesor en solitario.

A la hora de diseñar la aplicación hubo que centrarse en sí la división de nuestro escenario principal en dos partes bien diferenciadas, conllevaba también unas necesidades de comunicación diferentes para cada extremo, refiriéndonos a las prestaciones del software de videoconferencia a utilizar. Y efectivamente así fue, había que diseñar la videoconferencia de modo que cada extremo pudiese ejecutar unas funcionalidades distintas de las que ejecutaría el extremo opuesto.

Hubo que estudiar las necesidades y las facilidades que había que proporcionar a cada uno de los extremos. Se prestó, por tanto, especial atención a las características que diferencian a cada uno de los emplazamientos, pudiendo deducir así los requisitos que debía cumplir “Teletrófono” en cada extremo de la comunicación.

- El profesor en el extremo remoto necesita enviar tanto vídeo como audio al lado local.
- Por el contrario, desde el lado local no nos interesa enviar la imagen que capta la webcam en ese lado, pues no hay necesidad de que el lado remoto reciba información visual del aula.
- Sin embargo, sí nos interesa que el lado local sea capaz de enviar audio al lado remoto.
- Volviendo al lado remoto, sería interesante que el profesor en ese extremo pudiese visualizarse en su pantalla a sí mismo.

Con los anteriores puntos se deja bien definido el primer esbozo acerca del diseño de la aplicación y se pueden ver claramente las distintas necesidades de cada uno de los extremos.

Como hemos comentado en anteriores apartados, también se dispone de una herramienta de inserción de rótulos y de una herramienta de grabación de la sesión. Ambas herramientas se decidieron implementar únicamente en el extremo local. Esta decisión se tomó pensando en la comodidad de los dos actores principales de la videoconferencia, los dos profesores. Debemos tener en cuenta, que el profesor situado en el extremo remoto deberá preocuparse durante la videoconferencia de sostener el equipo físico en el que ha lanzado la aplicación (por defecto dijimos que sería un ordenador portátil), deberá ir apuntando la cámara web hacia aquello que le interesa mostrar en cada momento a los alumnos, deberá ir explicando todo aquello que está mostrando y además es probable que esté de pie o deba moverse de un lado a otro. Todas estas tareas que el profesor en remoto debe realizar sirven como motivo para situar las herramientas de inserción de subtítulos y de grabación en el lado local y liberar de carga trabajo al profesor en remoto. Por tanto, las funciones que deberá realizar el profesor en el aula durante la videoconferencia serán: transmitir dudas de los alumnos al profesor en el otro extremo y grabar la sesión e introducir subtítulos si lo desea.

Para llevar a cabo el diseño del resto de funcionalidades del programa, como son el reproductor y el editor de vídeo, no fue necesario distinguir entre lado local y lado remoto. A estas herramientas se puede acceder directamente desde el menú principal de la aplicación, que es común para cualquier extremo de la comunicación.

La aplicación, por tanto, es la misma en ambos extremos, lo que ocurre es que si se desea llevar a cabo una videoconferencia, hay que tener en cuenta qué lado va a jugar el papel de extremo local y qué lado va a jugar el papel de extremo remoto. Para llevar a cabo esta decisión habrá que tener muy en cuenta cómo se han definido en este documento cada uno de los lados. Una de las características claves que deberían determinar el papel de un extremo sería la movilidad. El lado de la comunicación que en



este documento se ha definido como “extremo remoto” se caracteriza por la posibilidad de movilidad e incluso movimiento del profesor. El lado de la comunicación que en este documento se ha definido como “extremo local” se caracteriza por ser un lugar fijo determinado que no requiere movilidad ni movimiento. Una vez asignados los roles, en función del extremo que se sea, se deberá ejecutar un icono u otro en el menú principal de la aplicación.

En la figura 4.4 se muestra un esquema general del diseño de la aplicación, teniendo en cuenta únicamente la parte de la aplicación que lleva a cabo la videoconferencia (es decir, excluyendo el reproductor y el editor de subtítulos) y suponiendo que desde el extremo local se está grabando la sesión.

En posteriores apartados se explicarán todos los detalles de implementación. Allí encontraremos los motivos por los que, por ejemplo, se ha decidido guardar el audio y vídeo que llega desde remoto a local en dos archivos diferentes, o por qué se han escogido esos formatos contenedores.

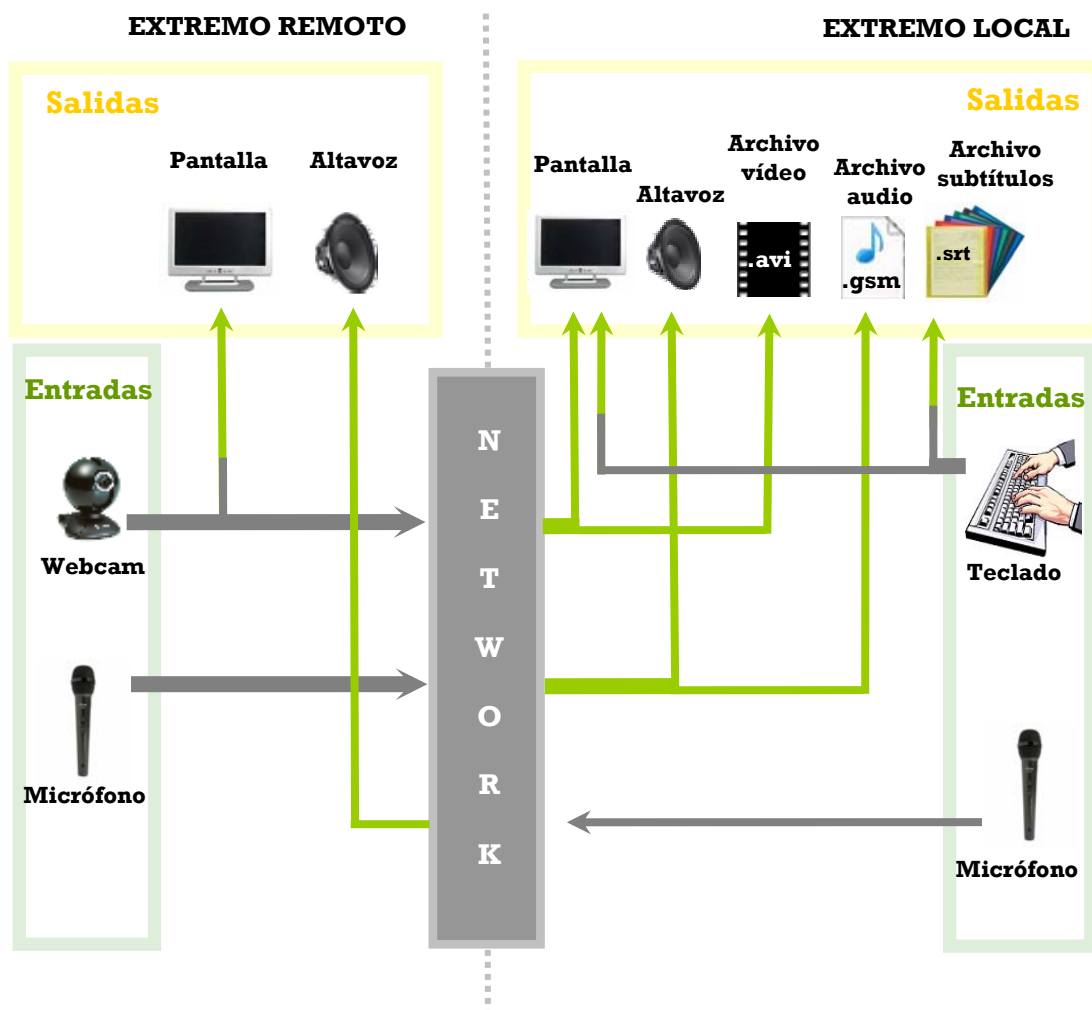


Figura 4.4. Esquema de videoconferencia a través de “Teletrófono” con grabación de la sesión.

#### 4.2.2.1. INTRODUCCIÓN A JAVA MEDIA FRAMEWORK.

##### Introducción

El lenguaje escogido para desarrollar la aplicación de videoconferencia ha sido Java. La primera versión de este lenguaje de programación, creado por ‘*Sun Microsystems*’, se publicó en mayo de 1995. Desde su introducción, esta plataforma ha sido una de las tecnologías nuevas que más rápidamente han sido adoptadas por la industria. Los principales proveedores de plataformas de computación han integrado la tecnología Java como un componente esencial de sus productos. Es, por tanto, un lenguaje de programación que ha alcanzado gran éxito y difusión y que está en constante evolución.

El éxito de este lenguaje se comprueba, por ejemplo, a través de un índice que elabora mensualmente la empresa “*Tiobe Software*” que mide la popularidad, a nivel mundial, de los principales lenguajes de programación. Si consultamos sus informes y gráficas ([www.tiobe.com](http://www.tiobe.com)) podemos ver como Java se mantiene a lo largo de los años como la plataforma de desarrollo más popular, según estos estudios.

Mientras tanto, Java sigue creciendo y ganando cuota de mercado basado en una comunidad cada vez más extensa de programadores individuales, instituciones y empresas, que apoyan el software libre.

*Java Media Framework* (JMF) es una librería adicional que extiende la plataforma Java Estándar (Java SE) y que nos permite trabajar con flujos multimedia en nuestras aplicaciones Java. JMF será, por tanto, una librería indispensable para desarrollar nuestra aplicación de videoconferencia, pues en ella necesitaremos tratar constantemente, tal y como hemos visto en el diseño, con flujos de audio y vídeo.

Sin embargo, esta librería de Java no cuenta exactamente con demasiada popularidad. Nace en 1997 de la mano de ‘*Sun Microsystems*’, ‘*Silicon Graphics*’ e ‘*Intel*’. Pero pronto éstos dos últimos abandonaron el proceso de especificación de JMF. A finales de 1998 “IBM” se mostró interesado en JMF, poco después se unió a ‘*Sun*’ y se publicó la versión JMF 1.1. Aunque se trata de una herramienta muy útil, ésta librería viene sufriendo durante casi una década una llamativa falta de actualizaciones. JMF parece no recibir esfuerzo alguno de mantenimiento por parte de ‘*Sun*’, como ejemplo, podemos comprobar que la API no se ha mejorado desde 2001.

Uno de los propósitos de este proyecto será evaluar *Java Media Framework* como biblioteca fundamental para poder implementar una videoconferencia educativa a través de Internet. Nada mejor que desarrollar una aplicación como “Teletrófono” mediante Java y JMF para poder evaluar apropiadamente la idoneidad de este paquete a la hora de implementar programas que trabajen con flujos multimedia. La dinámica de trabajo que se va a seguir consistirá en ir implementando la aplicación desde cero, sin haber usado anteriormente la biblioteca de JMF, e ir explicando paso a paso los problemas que nos hemos ido encontrando y cómo los hemos ido solucionando. De este modo descubriremos hasta dónde puede llegar JMF tal y como está actualmente, veremos el grado de dificultad que conlleva desarrollar código con esta biblioteca y averiguaremos qué tipo de problemas nos puede dar utilizar este lenguaje.

Pero antes de comenzar de lleno con la implementación de “Teletrófono” es necesario entender qué es JMF y como funciona. Veremos por tanto a continuación, un resumen sobre éste paquete Java. Toda la información que aparece en este apartado ha sido extraída de la “*Java Media Framework API Guide*” que podemos encontrar en la página web: <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/>

## Definición

*Java Media Framework* (entorno multimedia de trabajo de Java) proporciona la posibilidad de trabajar con datos multimedia en programas desarrollados con Java. JMF especifica un arquitectura para capturar, reproducir, almacenar o transmitir/recibir datos basados en el tiempo como vídeo o audio. Algunas clases de JMF trabajan con el protocolo RTP (Real-time Transport Protocol) que permite la transmisión y recepción de datos en tiempo real a través de una red.

## Conceptos Básicos

Una aplicación multimedia es aquella que produce, reproduce, procesa o maneja uno o varios contenidos multimedia. A su vez un contenido multimedia es aquel que está compuesto de diversos “medios”, como pueden ser audio, vídeo, texto, etc. Decimos que un contenido multimedia está basado en el tiempo en tanto que cada uno de sus medios cambia significativamente con él.

Cada uno de los medios de los que se compone un contenido multimedia se denomina pista o *track*. Por ejemplo un contenido multimedia correspondiente a una videoconferencia puede contener una pista de audio y otra de video. A su vez cada pista posee un formato que define como están estructurados los datos que forman parte de ella. Más adelante se especificarán los formatos soportados por JMF para los diferentes tipos de medios.

## Clases básicas en JMF

•**Managers:** JMF consiste principalmente en interfaces que definen el comportamiento y la interacción con los objetos usados para capturar, procesar, y presentar los datos multimedia. Usando objetos intermediarios llamados *Managers*, JMF hace fácil integrar las nuevas implementaciones de interfaces clave que pueden ser utilizadas a la perfección con las clases existentes. JMF usa cuatro *Managers*: *Manager*, *PackageManager*, *CaptureDeviceManager* y *PlugInManager*. En nuestro programa de videoconferencia haremos uso de dos de ellos:

- *Manager*: es quien construye los *Players*, *Processors*, *DataSources* y *Datasinks*.
- *CaptureDeviceManager*: quien mantiene un registro de los dispositivos de captura disponibles.

• **DataSource:** Una *DataSource* encapsula tanto la ubicación de la fuente multimedia como el protocolo necesario para transmitirla. Una *DataSource* maneja un conjunto de *SourceStream* objects. Las *Datasources* se dividen en dos clases dependiendo de quien inicie la transferencia de ésta: *PushDataSource* y *PullDataSources*. Aparte de las *DataSource*s tradicionales JMF define dos tipos especiales de *DataSource*: las “clonables” (*cloneables*) y las fusionadas (*merged*).

Una *DataSource cloneable* se puede usar para crear clones de una *DataSource*. Para crear este tipo especial debemos llamar al método del *Manager* *createCloneableDataSource*, pasándole la *DataSource* que queremos clonar. A partir de entonces sólo deberemos interactuar con la *DataSource* “clonable” y sus clones. Las *DataSource*s “clonables” implementan la interfaz *SourceCloneable* que define el método *createClone*. A partir de este método se pueden crear multitud de clones de la *DataSource* original.

Una *MergingDataSource* se puede usar para combinar los *SourceStreams* de varias *DataSource*s en una sola *DataSource*. Para construir una *MergingDataSource* se debe llamar al método del *Manager* *createMergingDataSource* pasándole un array que contenga las *DataSource*s a mezclar.

• **Players:** Un *Player* procesa un flujo de datos multimedia de entrada y lo reproduce. Se usa una *DataSource* para entregar el flujo de entrada al *Player*.

• **Processors :** Los *Processors* también se pueden usar para reproducir datos multimedia. Un *Processor* puede enviar sus datos de salida a un dispositivo que presente los datos o a una *DataSource*. En este último caso la *DataSource* de salida podrá ser pasada a un *Player*, a otro *Processor* o como entrada de un *DataSink*. Un *Processor* es simplemente un tipo especializado de *Player* que además permite controlar el procesamiento de los datos de entrada.

• **DataSinks:** Un *DataSink* se usa para leer datos de una *DataSource* y volcarlos en un destino concreto que no sea un dispositivo de presentación. Normalmente se usa para escribir datos en un archivo o a través de una red.

### **Presentar medios basados en tiempo con JMF (Player)**

Para reproducir medios basados en el tiempo como vídeo o audio con JMF se utiliza un *Player*. Un *Player* es un reproductor al que se le puede añadir un panel de control a través del cual se controla la reproducción. Si tenemos varios medios o pistas que reproducir se necesita un *Player* para cada una de ellas.

Para reproducir un medio se necesita construir un *Player* para ese flujo, configurarlo y prepararlo para después comenzar la reproducción.

- Crear un *Player*:

Se puede crear un *Player* a través del *Manager*, con la sentencia *createPlayer*. Para mostrar el *Player* habrá que obtener determinados objetos componentes de éste y añadirlos a la ventana de nuestro programa.

Un *Player* puede estar en seis estados distintos: *Unrealized*, *Realizing*, *Realized*, *Prefetching*, *Prefetched*, *Started*.

La mayor parte de los métodos que pueden ser llamados desde un *Player* requieren que éste esté en el estado *Realized*. Una forma de garantizar esto es llamar desde el *Manager* al método *createRealizedPlayer*, que crea el *Player* y espera hasta que éste esté en el estado *Realized*. En el siguiente esquema podemos ver los diferentes estados en los que se puede encontrar un *Player* y los métodos que pueden realizar las transiciones entre los distintos estados:

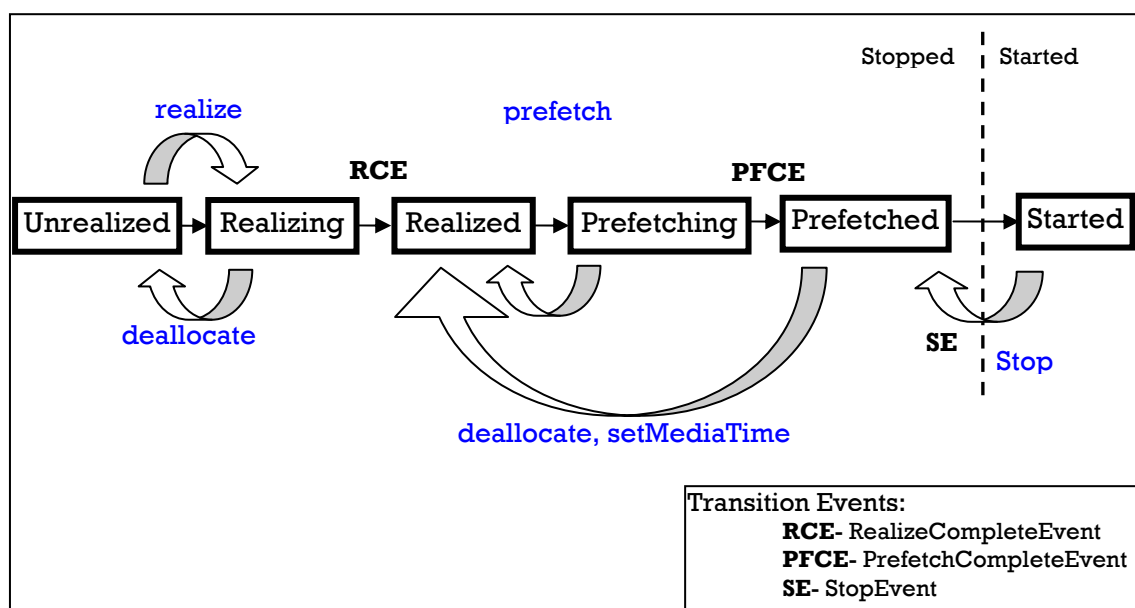


Figura 4.5. JMF: Estados de un *Player*.

(Fuente: *JMF API Guide*, pág. 26)

#### ● Reproducir flujos RTP multimedia:

En principio, los flujos RTP se pueden reproducir a través de un JMF *Player* construido a través del *Manager* usando un *MediaLocator* que tenga los parámetros de la sesión RTP. Sin embargo, si se usa este modo de reproducción del flujo, sólo el primer flujo RTP que se detecte en la sesión podrá ser presentado. Más adelante se explicará cómo podemos reproducir varios flujos RTP.

#### **Procesar medios basados en tiempo con JMF (*Processor*)**

Un *Processor* puede usarse como un *Player* programable que te permite controlar el procesamiento de los datos, incluyendo la codificación y multiplexación de datos capturados. Se puede controlar qué tipo de procesamiento está llevando a cabo un *Processor* de diversas formas, algunas de ellas son:

- usando un *ProcessorModel* para construir un *Processor* con ciertas características de entrada y salida,
- usando el método *TrackControl setFormat* para especificar qué conversiones de formato se van a llevar a cabo sobre cada una de las pistas,
- usando el método *setOutputContentDescriptor* de un *Processor* para especificar el formato de los datos multiplexados de salida, etc.

Un *Processor* tiene dos estados más que un *Player*, “*Configuring*” y “*Configured*”, los cuales tienen lugar antes de que el *Processor* entre en el estado “*Realizing*”. Como dijimos, un *Processor* es un tipo de *Player*, así que las restricciones en los métodos que pueden ser llamados también se aplican a los *Processors*. Algunos de los métodos específicos de los *Processors* sólo se pueden llamar en determinados estados.

En el siguiente esquema podemos ver los diferentes estados en los que se puede encontrar un *Processor* y los métodos que pueden realizar las transiciones entre los distintos estados:

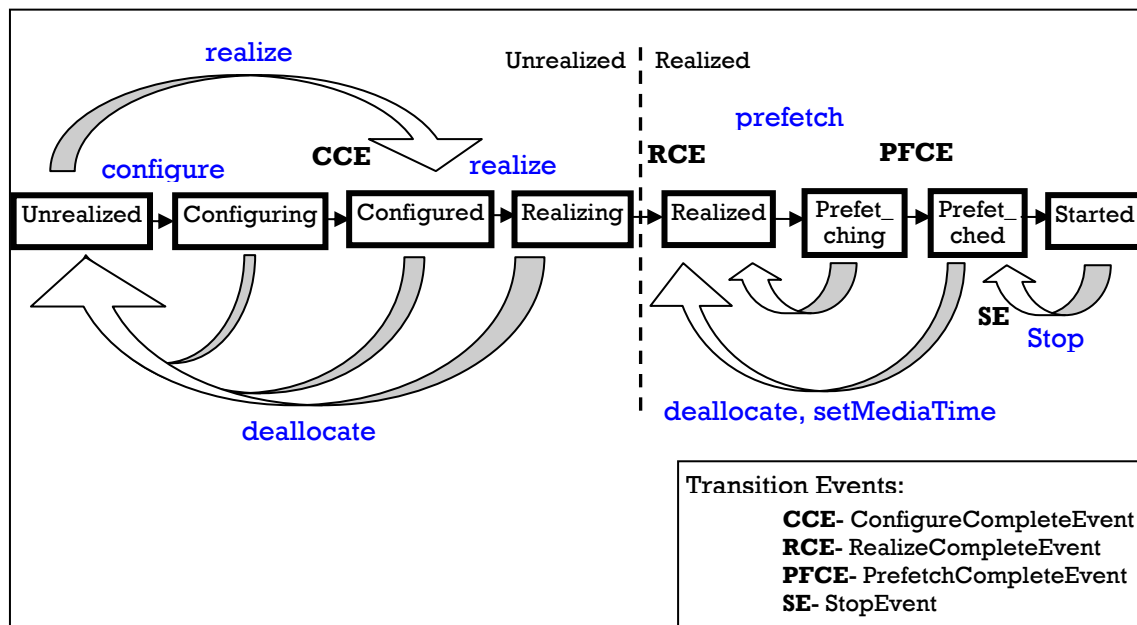


Figura 4.6. JMF: Estados de un *Processor*.

(Fuente: *JMF API Guide*, pág. 34)

- Para seleccionar los “*plug-ins*” que se van a usar para procesar cada pista del flujo de datos:

En primer lugar habrá que, llamar al método *PlugInManager.getPlugInList* para determinar los “*plug-ins*” disponibles. El *PlugInManager* devolverá una lista con aquellos que cuadren con el tipo de *plug-in* escogido y los formatos de entrada y salida. En segundo lugar deberemos llamar al método *getTrackControls* del *Processor* para obtener el objeto *TrackControl* de cada pista del flujo multimedia. Para ello el *Processor* debe estar en estado “*Configured*”. Por último llamaremos al método *setCodecChain* o *setRenderer* de *TrackControl* para especificar los “*plug-ins*” que se quieren usar para cada pista.

Para controlar la codificación que se está llevando a cabo en una pista por un determinado códec debemos serviros de los controles de códec asociados a esa pista.

Dichos controles de códec los obtenemos llamando al método *getControls* de *TrackControl*.

- Para convertir datos multimedia de un formato a otro:

Se puede seleccionar el formato de una pista determinada a través del *TrackControl* de esa pista. En primer lugar llamaríamos al método del *Processor* *getTrackControls* para obtener los controles para cada pista. Luego usaríamos el método *setFormat* de *TrackControl*, para seleccionar el formato concreto al que se quiere convertir la pista.

- Para especificar el destino del flujo de datos multimedia:

Se puede seleccionar un destino concreto para el flujo de datos seleccionando un *Renderer* particular para un track, a través de su *TrackControl* o tomando la salida de un *Processor* como entrada a un determinado *DataSink*. También podríamos usar la salida del *Processor* como entrada de un *Player* o *Processor* que tenga un destino diferente. Para obtener los datos de salida de un *Processor* llamamos al método *getDataOutput*, que nos devuelve una *DataSource*.

### **Capturar medios basados en tiempo con JMF**

JMF puede usarse para capturar datos basados en el tiempo de un dispositivo de captura como un micrófono o una cámara de vídeo. Los datos capturados pueden ser procesados, reproducidos o almacenados para un uso futuro.

Para acceder a los dispositivos de captura de los que disponemos debemos acudir al *CaptureDeviceManager*. El *CaptureDeviceManager* es el registro central para todos los dispositivos de captura accesibles a JMF. Podemos obtener una lista de los dispositivos disponibles en nuestro equipo llamando al método *CaptureDeviceManager.getDeviceList*. Cada dispositivo de captura se representa a través de un objeto *CaptureDeviceInfo*.

Para capturar datos de un dispositivo necesitamos obtener el *MediaLocator* de su objeto *CaptureDeviceInfo*. Se puede usar dicho *MediaLocator* para construir un *Player* o *Processor* directamente o usar el *MediaLocator* para construir una *DataSource* que se puede usar como entrada de un *Player* o un *Processor*. Para iniciar el proceso de captura bastaría con comenzar (*start*) el *Player* o el *Processor*.

- Controlando el proceso de captura:

Un dispositivo de captura tiene normalmente implementados un conjunto de atributos que pueden usarse para controlar el dispositivo. Hay dos tipos de control definidos en este ámbito: *PortControl* y *MonitorControl*. Para acceder a estos controles se debe llamar al método *getControl* de la *DataSource* capturada y pasarle en nombre del control que deseamos. Un *PortControl* nos permitirá seleccionar el puerto desde el cual serán capturados los datos. Un *MonitorControl* nos permitirá mostrar la monitorización de los dispositivos de captura. Como los demás objetos de control, si hay un componente visual que corresponde a alguno de estos controles, lo podemos obtener llamando al método *getControlComponent*. Añadir el componente a la ventana de la aplicación permitirá a los usuarios interactuar con los controles de captura.

- Almacenar los datos capturados:

Si se desea guardar los datos capturados en un archivo, se necesitará usar un *Processor* en lugar de un *Player*. Se deberá usar un *DataSink* para leer los datos desde la *DataSource* de salida del *Processor* y enviar los datos a un archivo. Por tanto, en primer lugar se deberá obtener la *DataSource* de salida del *Processor*. Después se deberá crear un *DataSink* mediante la sentencia *Manager.createDataSink*, pasándole la citada *DataSource* y un *MediaLocator* que especifique la ubicación del archivo en el que queremos escribir. Seguidamente deberemos abrir el *DataSink* y llamar al método *start* del *DataSink* y el *Processor*.

## Protocolo RTP (Real-Time Transport Protocol)

Para enviar o recibir una retransmisión en directo o llevar a cabo una videoconferencia sobre Internet es necesario ser capaz de recibir y transmitir flujos multimedia en tiempo real.

Transmitir datos multimedia a través de la red en tiempo real requiere un alto rendimiento. Es más fácil compensar la pérdida de datos que compensar altos retardos en la recepción de los datos. Esta situación es muy diferente a la de transmitir datos estáticos, por ejemplo, en un fichero, donde lo más importante es que todos los datos lleguen a su destino. Consecuentemente los protocolos que se usan para datos estáticos no funcionan bien para transmitir datos multimedia en tiempo real.

Los protocolos HTTP (*HyperText Transfer Protocol*) y FTP (*File Transfer Protocol*) se basan en el protocolo TCP (*Transmission Control Protocol*). TCP es un protocolo de la capa de transporte diseñado para transmitir datos de forma fiable sobre redes con bajo ancho de banda y alta tasa de error. Cuando un paquete se pierde o daña, se retransmite. La sobrecarga que permite garantizar la transferencia de datos fiable disminuye la tasa de transmisión general. Por este motivo, los protocolos subyacentes distintos de TCP se usan típicamente para datos multimedia en tiempo real.

Uno de estos protocolos comúnmente usados es UDP (*User Datagram Protocol*). UDP es un protocolo no fiable, es decir, no garantiza que cada uno de los paquetes alcance su destino. Tampoco hay garantías de que los paquetes lleguen en el orden en el que fueron enviados. El receptor debe ser capaz de compensar las pérdidas de datos, los paquetes duplicados y los desordenados. Como TCP, UDP es un protocolo de la capa de transporte.

El estándar de Internet para transportar datos en tiempo real como audio y vídeo es el *Real-Time Transport Protocol* (RTP). RTP provee servicios para la transmisión extremo a extremo en una red de datos en tiempo real. RTP es independiente de la red y el protocolo de transporte usado, aunque normalmente se usa sobre UDP.

RTP puede usarse sobre servicios de red *unicast* o *multicast*. En un servicio de red *unicast*, se mandan copias separadas desde el origen a cada uno de los destinos. En un servicio *multicast* se mandan los datos desde origen sólo una vez, siendo la red la responsable de hacer llegar esos datos a cada uno de los destinos.

RTP permite identificar el tipo de datos que están siendo transmitidos, determinar el orden en el que los paquetes deberían presentarse y sincronizar flujos de datos multimedia provenientes de diferentes fuentes.



En RTP no se garantiza que los paquetes de datos lleguen en el mismo orden en el que fueron enviados, de hecho, ni siquiera se garantiza del todo que lleguen. Es tarea del receptor reconstruir la secuencia de paquetes y detectar la pérdida de paquetes, usando la información que contiene la cabecera de estos.

Aunque RTP no provee mecanismo alguno para asegurar la entrega a tiempo o proporcionar otras garantías de calidad de servicio, ésta se aumenta gracias un protocolo de control (RTCP) que permite monitorizar la calidad de la distribución de los datos. RTCP también proporciona mecanismos de control e identificación.

### Arquitectura RTP

Una sesión RTP es la relación entre un conjunto de aplicaciones que se comunican a través de RTP. Una sesión se identifica por una dirección de red y un par de puertos. Uno de los puertos se usa para transmitir los datos multimedia y otro para transmitir los datos de control (RTCP).

Un participante es una máquina, *host* o usuario que participa en la sesión. La participación en una sesión puede consistir en la recepción pasiva de datos (receptor), la transmisión activa de datos (transmisor), o ambas.

Cada tipo de dato se transmite en una sesión diferente. Por ejemplo, si en una videoconferencia se usa tanto audio como vídeo, usaremos una sesión para transmitir el audio y otra sesión diferente para transmitir el vídeo. Esto permite a los participantes elegir el tipo de datos que desean recibir, por ejemplo, si alguien tuviese una conexión de red con bajo ancho de banda quizá querría recibir solamente la parte de audio de una videoconferencia.

Los datos en una sesión se transmiten como series de paquetes. Una serie de paquetes de datos que tienen su origen en una fuente de datos determinada se denomina “RTP Stream”, es decir, flujo RTP. Cada uno de los paquetes de datos de un flujo RTP se compone de una cabecera estructurada (*header*) y los datos en sí (*payload*).

### Paquetes de control

Los paquetes de control de datos (RTCP) se mandan periódicamente a todos los participantes en la sesión. Los paquetes RTCP pueden contener información acerca de la fuente de la que provienen los datos que están siendo transmitidos en el puerto, la calidad del servicio y estadísticas referentes a estos datos.

Hay varios tipos de paquetes RTCP: puede ser un informe del remitente (*Sender Report*), un informe del destinatario (*Receiver Report*), una descripción de la fuente de datos (*Source Description*), un mensaje de desconexión (*Bye*) o un mensaje específico de una aplicación.

Todos los participantes en una sesión envían paquetes RTCP. Los paquetes RTCP se envían como paquetes compuestos que contienen al menos dos paquetes: un paquete de informe y un paquete de descripción. El primer paquete en un paquete RTCP compuesto debe ser un informe, incluso cuando aún ni siquiera se han enviado o recibido datos. En ese caso se envía un informe de destinatario vacío. Todos los paquetes compuestos

RTCP deben incluir un elemento de descripción de la fuente (SDES) que contiene, entre otros muchos campos, el nombre canónico (CNAME) que identifica la fuente.

### Protocolo RTP (Real-Time Transport Protocol) y Java Media Framework

JMF permite la reproducción y transmisión de flujos RTP a través de la API definida en los paquetes `javax.media.rtp`, `javax.media.rtp.event` y `javax.media.rtcp`. Podremos reproducir localmente flujos RTP recibidos, guardarlos en un archivo, o ambas cosas. Podremos hacer lo mismo con los flujos RTP que enviamos. Vemos un par de esquemas ejemplo:

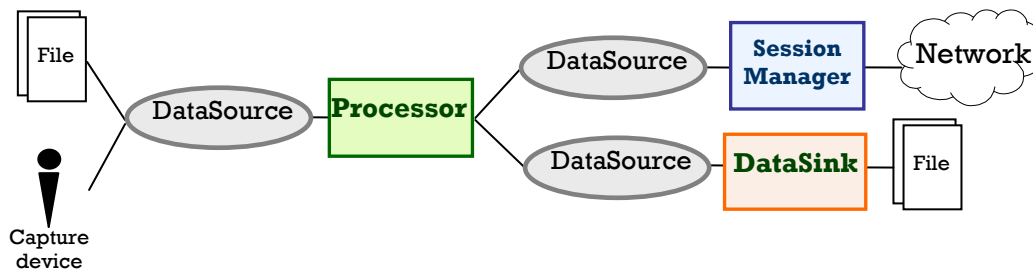


Figura 4.7. JMF: Transmisión RTP.

(Fuente: *JMF API Guide*, pág. 118)

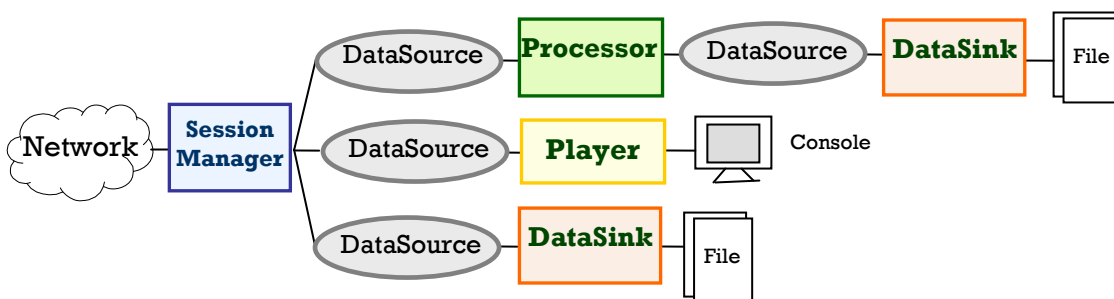


Figura 4.8. JMF: Recepción RTP.

(Fuente: *JMF API Guide*, pág. 117)

En JMF el *SessionManager* (*RTPManager*) es el objeto que se usa para coordinar una sesión RTP. Éste hace un seguimiento de los participantes de la sesión y de los flujos que se transmiten. El *SessionManager* mantiene el estado de la sesión, vista desde el participante local. De modo que este administrador de sesiones es en realidad una representación de la sesión RTP. El *SessionManager* también maneja el canal de control RTCP. La interfaz de este administrador de sesiones define métodos que permiten a una aplicación iniciar y comenzar a participar en una sesión, eliminar flujos individuales creados por la aplicación, y cerrar la sesión por completo. El administrador de la sesión

también maneja estadísticas sobre todos los paquetes RTP y RTCP enviados y recibidos en una sesión.

El *SessionManager*, tal y como hemos dicho, hace un seguimiento de los participantes en una sesión. Cada participante se representa con una instancia de una clase que implemente el interfaz *Participant*. Los administradores de la sesión crean un *Participant* siempre que llegue un paquete RTCP que contenga una descripción de fuente (SDES) con un nombre canónico (CNAME) que no haya sido visto antes en la sesión. Los participantes pueden ser activos o pasivos. Un participante puede ser el propietario de varios flujos, cada uno de los cuales se identifica por el SSRC (*Synchronization Source Identifier*) usado por la fuente del flujo.

El administrador de la sesión mantiene un objeto *RTPStream* para cada flujo de paquetes de datos en una sesión. Hay dos tipos de flujos RTP: *ReceiveStream* (representa un flujo que se ha recibido desde un participante remoto) y *SendStream* (representa un flujo de datos que se está enviando por la red). Cuando el administrador de la sesión detecta una nueva fuente de datos RTP, construye automáticamente un *ReceiveStream*. Para crear un nuevo *SendStream* hay que llamar al método *createSendStream* del *SessionManager*.

Existen eventos RTP que se usan para informar del estado de la sesión RTP y los flujos. Para recibir notificación de los eventos RTP hay que implementar el oyente RTP adecuado y registrarlo en el administrador de sesiones. Existen varios oyentes RTP: *SessionListener*, *SendStreamListener*, *ReceiveStreamListener* y *RemoteListener*. Cada uno de estos oyentes tiene asociados unos eventos determinados. Por ejemplo, el *SessionListener* tiene asociados los eventos *NewParticipantEvent* y *LocalCollisionEvent*.

Todos los datos específicos de RTP usan un formato de codificación específico de RTP tal y como se define en las clases *AudioFormat* y *VideoFormat*. Por ejemplo, los paquetes encapsulados en jpeg RTP tendrán una codificación *VideoFormat.JPEG\_RTP*.

La API RTP define un control específico para RTP, *RTPControl*. Este control lo implementan típicamente las *DataSources* específicas de RTP. Provee un mecanismo para asignar un determinado formato a una carga de datos (*payload*). *RTPControl* también dispone de métodos para acceder a las estadísticas de la sesión y obtener el formato de los datos actuales.

- La presentación de un flujo RTP de entrada se lleva a cabo mediante un *Player*. Para recibir y mostrar un único flujo RTP entrante, se puede usar un *MediaLocator* que describa la sesión con la que se quiere construir el *Player*. Un *MediaLocator* para una sesión RTP tendrá la forma: *rtp://address:port[:ssrc]/content-type/[ttl]*. El *Player* será construido y se conectará al primer flujo de la sesión. Si existen varios flujos en la sesión que quieras mostrar, entonces necesitaremos acudir al administrador de la sesión. Se puede recibir un aviso del administrador de la sesión cada vez que se añada un flujo a la sesión y así construir un *Player* para cada flujo.

## Recibir y Presentar flujos RTP con Java Media Framework

Los *Players* y *Processors* son en JMF los mecanismos encargados de presentar, capturar y convertir flujos de datos RTP. Un ejemplo de recepción de flujos RTP y su posterior presentación sería:

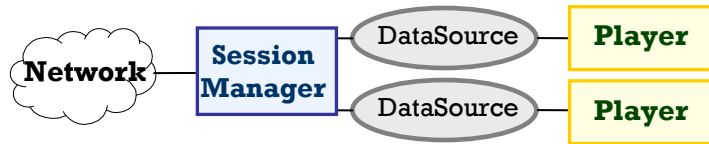


Figura 4.9. JMF: Recepción y presentación de un flujo RTP.

(Fuente: *JMF API Guide*, pág. 129)

Como vemos, se ha de usar un reproductor para cada uno de los flujos recibidos por el gestor de la sesión. Para construir el *Player* llamaremos al método del *Manager*: *CreatePlayer*. Podemos:

- usar un *MediaLocator* que contenga los parámetros de la sesión RTP y construir el reproductor mediante la sentencia: *Manager.createPlayer(MediaLocator)*,
- o bien, construir un reproductor para un flujo *ReceiveStream* en particular, obteniendo la *DataSource* del flujo y con la sentencia: *Manager.createPlayer(DataSource)*.

Sin embargo, si usamos un *MediaLocator* para construir el reproductor, tan sólo podremos presentar el primer flujo RTP detectado en la sesión. Si queremos reproducir varios flujos de una sesión necesitaremos, por tanto, usar el *SessionManager* y construir un reproductor para cada *ReceiveStream*.

### Transmitir flujos RTP con Java Media Framework

Para transmitir un flujo RTP usaremos un *Processor* para generar una *DataSource* codificada especialmente para RTP y construiremos un *SessionManager* o un *DataSink* para controlar la transmisión. La entrada del procesador pueden ser datos almacenados o capturados en vivo y en directo. Cuando creamos el *Processor*, para datos almacenados, podemos usar un *MediaLocator* que identifique el archivo. Para datos recién capturados, se usará como entrada al procesador la *DataSource* capturada.

Hay dos modos de transmitir flujos RTP:

- usar un *MediaLocator* que contenga los parámetro de la sesión RTP para construir un *DataSink* mediante la sentencia: *Manager.createDataSink*,
- o bien , usar un administrador de la sesión para crear flujos con el contenido y controlar la transmisión.

Si usamos un *MediaLocator* para construir el *DataSink* RTP, sólo podremos transmitir el primer flujo de la *DataSource*. Si queremos transmitir varios flujos RTP en una sesión o queremos monitorizar las estadísticas de la sesión, habrá que usar directamente un *SessionManager*.

Por tanto, independientemente de cómo escojamos transmitir el flujo RTP, necesitaremos: crear un *Processor* con la *DataSource* que representa los datos que queremos transmitir, configurar el procesador para que nos devuelva datos codificados en RTP y obtener la salida del *Processor* como una *DataSource*.

#### 4.2.2.2. DISEÑO BASADO EN JMF.

Tras haber explicado muy brevemente los principios y clases principales que guían a la biblioteca Java Media Framework, estamos en disposición de adentrarnos en el diseño de “Teletrófono”. En este apartado, trataremos de configurar un diseño para nuestra aplicación basándonos en los objetos con los que trabaja JMF. Ya en el apartado 4.2.3 será donde trataremos de implementar dicho esquema.

Para configurar el esquema que nos sirva de guía para comenzar a implementar nuestra aplicación debemos recordar la figura 4.4 de este documento. En ella aparecía un diseño informal de las funcionalidades de la herramienta de videoconferencia para ambos extremos de la comunicación.

Comencemos centrándonos en el extremo remoto. En dicho extremo se encontrará un profesor de que debemos capturar información tanto auditiva como visual. Deberemos, por tanto, disponer de dos dispositivos de captura: un micrófono y una cámara web. Deberemos acceder a estos dispositivos de captura a través de la clase *CaptureDeviceManager*, tal y como se indicaba en el anterior apartado. Una vez podamos acceder a esos dispositivos, deberemos crear un *MediaLocator* que haga referencia a cada uno de esos dispositivos y a través de éste crear ambas *DataSources*.

El siguiente paso es pensar qué queremos hacer con cada uno de estos flujos de datos. Si recordamos, queremos:

- mostrar en la propia pantalla del lado remoto, la imagen que está capturando su webcam,
- enviar al otro extremo dicha imagen que captura la webcam,
- y enviar también al otro extremo, el audio que capturamos con el micrófono.

Como vemos, el flujo de datos que obtenemos de la webcam queremos enviarlo a dos destinos distintos. Para visualizar ese flujo en nuestra pantalla, necesitaremos simplemente un *Player*; para enviar ese flujo al otro extremo de la comunicación deberemos usar varios elementos, pues necesitaremos establecer una conexión RTP con el otro extremo. En un principio podríamos pensar en utilizar la *DataSource* que obtenemos de la webcam, como entrada tanto del *Player* como del conjunto de elementos que necesitaremos previos a la transmisión RTP, pero eso no es posible con JMF.

Deberemos acudir a un tipo especial de *DataSource*, la ‘clonable’. Este tipo de *DataSource* nos permite crear un tipo de fuente de datos de la que podremos obtener copias exactas. Se trata, por tanto, de crear una *cloneable DataSource* con el flujo de vídeo que capta la cámara web. Pasaremos ésta fuente a un reproductor y, a partir de ella crearemos un clon que conformará el flujo de datos que trataremos de enviar al otro extremo.

Para crear una *DataSource* “clonable” deberemos llamar al método del Manager, *createCloneableDataSource* pasándole la fuente de datos original, en este caso, la que obtenemos directamente de la webcam. Las *DataSources* ‘clonables’ implementan la interfaz *SourceCloneable*, quien define un método, *createClone*, con el que se pueden crear todas las copias exactas que queramos de la fuente de datos original. A partir de

este momento sólo deberemos interactuar con la fuente ‘clonable’ y sus clones; la fuente original no deberá ser usada directamente.

De momento, entonces, tenemos clara la siguiente parte del diseño:

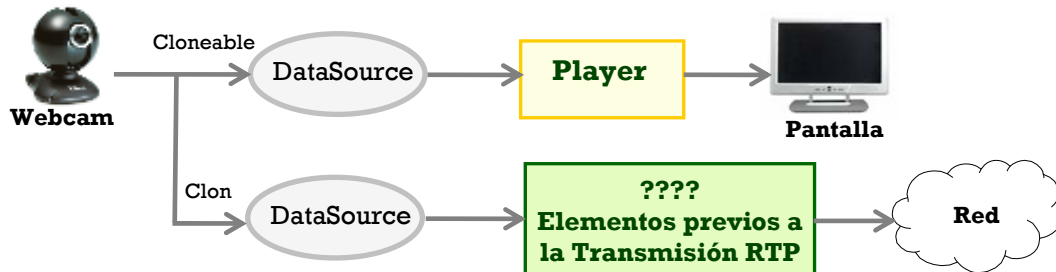


Figura 4.10. Diseño de la aplicación “Teletrófono”. Esquema ‘A’ lado remoto.

Para terminar de definir el anterior esquema nos falta por determinar los elementos que necesitamos para llevar a cabo la transmisión RTP. Consultando los manuales de JMF, nos encontramos que para transmitir vía RTP un flujo de datos, necesitaremos introducir previamente dicho flujo en un *Processor*. Este procesador será el encargado de convertir los datos de entrada en datos codificados especialmente para ser transmitidos sobre RTP. Habrá que establecer formatos RTP específicos para cada pista y especificar el descriptor del contenido de salida que deseamos. Los formatos de cada pista se establecen obteniendo el *TrackControl* de cada pista y llamando al método *setFormat* para especificar un formato RTP concreto. El formato de salida lo estableceremos con el método *setOutputContentDescriptor*. Si no se requiere ningún tipo de multiplexación en concreto se puede fijar este descriptor como “*ContentDescriptor.RAW*”. De todos modos, veremos más profundamente cuál es el modo exacto de configurar el *Processor* cuando describamos la implementación del programa.

Lo que sí debemos tener en cuenta en este instante es que los flujos de audio y de vídeo no deben entrelazarse. Cada flujo de datos deberá ser siempre transmitido en sesiones RTP diferentes.

¿Nos hace falta algún elemento adicional aparte del procesador?, pues nos hace falta un elemento esencial para las transmisiones RTP: el *SessionManager*. En realidad hay dos formas de realizar una transmisión RTP: la primera de ellas es realizar la transmisión a través de un *DataSink* construido mediante un *MediaLocator* con los parámetros de la sesión, la otra de las opciones es utilizar el *SessionManager* para enviar el flujo. Optaremos por la segunda de las opciones porque es la que más posibilidades nos va a proporcionar. Utilizar un *SessionManager* nos permite transmitir varios flujos RTP en una sesión así como monitorizar las estadísticas de la sesión.

Por tanto, una vez que el *Processor* ha terminado de establecer el formato de los datos, podemos obtener la *DataSource* de salida de este procesador a través del método *getDataOutput*. Esta fuente de datos de salida se conectará al *SessionManager* usando el método *createSendStream*.

El nuevo esquema para la captura, visualización y envío de la imagen del extremo remoto al local quedaría del siguiente modo:

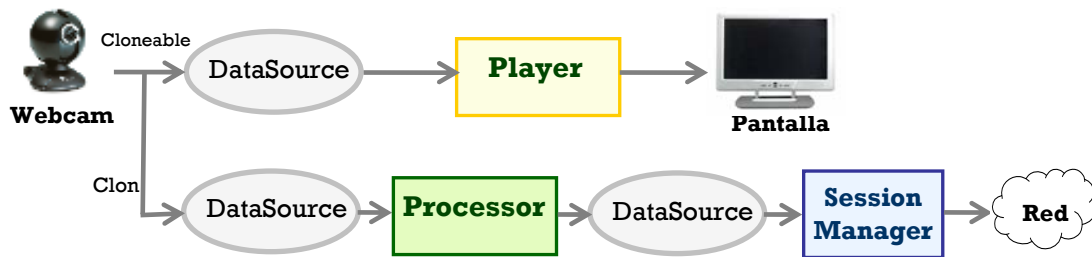


Figura 4.11. Diseño de la aplicación “Teletrófono”. Esquema ‘B’ lado remoto.

Una vez decidido el esquema a seguir con los datos que captura la cámara web, vamos a seguir el mismo proceso con el micrófono. Queremos que el audio que captura el micrófono en el lado remoto sea enviado al otro extremo de la comunicación.

Podríamos pensar que una buena idea sería aprovechar la sesión de vídeo establecida con el extremo local, para enviar también el audio. Pero eso no es posible. Si tenemos un flujo de audio y otro de vídeo, éstos siempre deberán enviarse a través de sesiones RTP diferentes. El audio lo enviaremos por una sesión RTP y el vídeo por otra sesión RTP.

Entonces, para llevar a cabo el envío del audio captado por el micrófono del extremo que nos ocupa deberemos seguir exactamente el mismo esquema que hemos seguido para el envío del vídeo a través de RTP. Deberemos crear, en primer lugar, un procesador que codifique los datos capturados por el micrófono a un formato entendible por el protocolo RTP. Una vez se halla realizado esta transformación deberemos obtener la salida del *Processor* y proporcionar dichos datos de salida al *SessionManager*. Éste último será el encargado de hacer llegar los datos de interés al otro extremo a través de la red.

Partiendo de los dispositivos de captura protagonistas en el extremo remoto, hemos diseñado parte del esquema que debe seguir la aplicación “Teletrófono”. Ahora hemos de repasar los dispositivos de salida protagonistas en el mismo extremo, para comprobar si nos falta algún proceso por diseñar.

En la figura 4.4 aparecían como dispositivos de salida del extremo remoto una pantalla y un altavoz. La pantalla nos sirve para ir mostrando la imagen que va capturando la propia webcam de este lado, proceso que ya hemos diseñado. El altavoz nos servía para poder reproducir el flujo de audio que nos debe llegar desde el extremo local. Este último proceso es el que vamos a considerar a continuación.

La pregunta en este momento es: ¿cómo recibir un flujo de audio a través de una conexión RTP y reproducirlo? En JMF la recepción de flujos RTP debe gestionarla de nuevo el *SessionManager*. Aparte de este administrador, deberemos tener otro bloque que nos permita reproducir en el altavoz esos flujos de audio que está recogiendo el *SessionManager*. Construiremos, por tanto, un *Player* para un determinado *ReceiveStream*. Para ello deberemos obtener la *DataSource* del flujo y ejecutar la sentencia: *Manager.createPlayer(DataSource)*.

El esquema de diseño, teniendo en cuenta los procesos de comunicación en el extremo remoto en los que el micrófono y el altavoz son los protagonistas, quedaría como sigue:

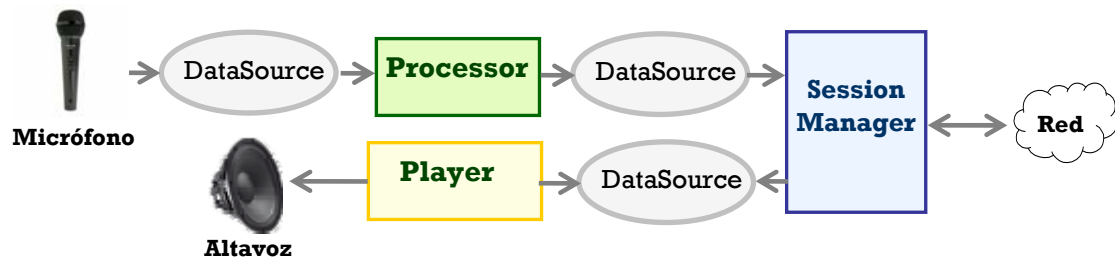


Figura 4.12. Diseño de la aplicación “Teletrófono”. Esquema ‘C’ lado remoto.

Tras habernos asegurado de que no queda ningún proceso por diseñar en este extremo donde se sitúa el profesor que se ha trasladado a un lugar arbitrario para impartir clase desde allí a sus alumnos, sólo nos queda unir los esquemas obtenidos. Finalmente el esquema que contiene el diseño del lado remoto será el siguiente:

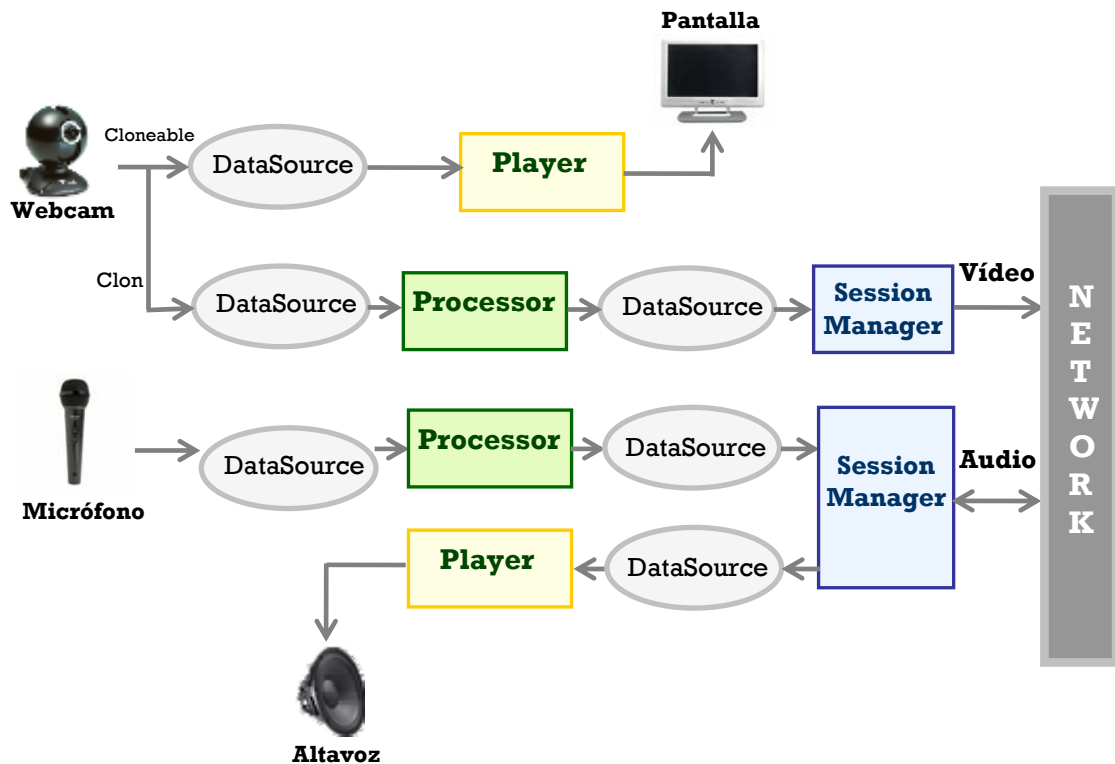


Figura 4.13. Diseño de la aplicación “Teletrófono”. Esquema ‘D’ lado remoto.

Habiendo finalizado el diseño del extremo remoto, centrémonos ahora en el extremo local. En dicho extremo se encontrará un profesor en un aula acompañado por un grupo de alumnos. La única información multimedia de que debemos capturar en este lado, es información auditiva. Deberemos por tanto, en un principio, disponer de un único dispositivo de captura: un micrófono. Deberemos acceder a este dispositivo de captura a través de la clase *CaptureDeviceManager*, tal y como se hacía en el otro extremo de la comunicación. Una vez accedamos a este dispositivo crearemos una *DataSource* a través de un *MediaLocator* que haga referencia al micrófono.



El siguiente paso es pensar qué queremos hacer con este flujo de datos. Si recordamos, queremos hacer exactamente lo mismo que en el otro extremo, en el caso del audio. Deseamos enviar al lado remoto el audio que capturamos en este lado con el micrófono. Del mismo modo, también desearemos reproducir en un altavoz, el flujo de audio que nos llega desde el extremo remoto. En este ámbito, por tanto, el esquema de diseño en ambos extremos es idéntico:

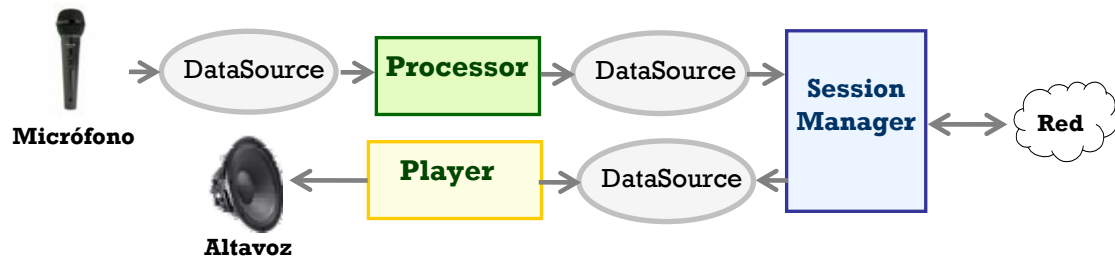


Figura 4.14. Diseño de la aplicación “Teletrófono”. Esquema ‘A’ lado local.

Ahora debemos pensar qué otras funcionalidades debía implementar este lado de la comunicación. Si recordamos, son las siguientes:

- mostrar por pantalla la imagen que está capturando la webcam situada en el otro extremo,
- guardar en un archivo de vídeo tanto el audio como el vídeo que se recibe desde el lado remoto,
- ir introduciendo subtítulos a la imagen recibida, a través del teclado. Dichos subtítulos deberán ir apareciendo en pantalla y quedar almacenados en un archivo de subtítulos estándar (por ejemplo, SRT).

Comencemos por el principio. Para mostrar por pantalla en el lado local la imagen que capta la cámara web del profesor situado en un entorno remoto, primero debemos establecer una sesión RTP a través de la cual recibir ese flujo de vídeo. Una vez más necesitaremos servrnos, por tanto, de un *SessionManager*. Este administrador de la sesión deberá ser capaz de detectar el flujo de vídeo de entrada y recogerlo. Para poder reproducir este flujo en nuestra pantalla deberemos crear un *Player* para ese *ReceiveStream*. Para ello deberemos obtener la *DataSource* del flujo y ejecutar la sentencia: *Manager.createPlayer(DataSource)*.



Figura 4.15. Diseño de la aplicación “Teletrófono”. Esquema ‘B’ lado local.

Otro de los requisitos de la aplicación en el lado que nos ocupa, es guardar en un archivo de vídeo tanto el flujo de vídeo como el de audio que se recibe desde el extremo remoto. Las soluciones posibles son varias. Sabemos que con RTP no podemos hacer varias cosas (procesar, reproducir, guardar, enviar, etc.) con una única DataSource, sino que se debe crear una DataSource “clonable” a partir de la cual crearemos copias idénticas. Mediante este sistema sí es posible llevar a cabo diferentes procesos con el “mismo” flujo de datos. Actualmente, en el punto que nos encontramos del diseño, estamos cogiendo el flujo de audio que nos llega desde remoto y reproduciéndolo en el

altavoz; y estamos cogiendo el flujo de vídeo que nos llega desde remoto y estamos visualizándolo en la pantalla. Como ahora queremos guardar estos flujos en un archivo, del modo que estamos planteando la situación, la solución más sencilla que se nos ocurre pasaría por clonar dichos flujos (tanto el de vídeo como el de audio) y almacenar en un archivo de vídeo dichos clones. Siempre teniendo en cuenta, que antes hay que construir con la *DataSource* original, una *DataSource* “clonable”.

Pero hay otro aspecto a solucionar. Sabemos que queremos obtener un único archivo de vídeo a partir de un flujo de audio y otro de vídeo. Para ello JMF nos da una solución, las fuentes de datos fusionadas, llamadas *MergingDataSources*. Una fuentes de datos de este tipo, se usa para combinar diferentes flujos provenientes de otras *DataSources* en una única *DataSource*. Esto permite administrar desde un sólo punto un conjunto de *DataSources*. Cuando los métodos de parar, comenzar, conectar o desconectar de llaman sobre la *MergingDataSource*, estas llamadas se propagan a las *DataSources* fusionadas. Como veremos en la implementación de “Teletrófono”, para construir una fuente de datos de esta clase, basta con llamar al método *Manager.createMergingDataSource* y pasarle un array que contenga las fuentes de datos que queremos mezclar. La duración de la *DataSource* fusionada será el máximo de entre las duraciones de cada objeto *DataSource*. Para que puedan ser mezcladas, todas las *DataSources* del array deberán ser del mismo tipo; por ejemplo, no se puede mezclar una *PullDataSource* con una *PushDataSource*.

Una fuente de datos pertenecerá a una categoría u otra de las que acabamos de citar, dependiendo de cómo de haya iniciado la transferencia de los datos. En una *PullDataSource* el cliente inicia la transferencia de los datos y controla los flujos de datos de estas fuentes. En una *PushDataSource* es el servidor quien inicia la transferencia y controla los flujos. En nuestro caso ambos flujos, tanto el de vídeo como el de audio, provienen de la red (en realidad del extremo remoto) y llegan al extremo local, por lo que han de ser del mismo tipo y no deberíamos tener problema alguno al tratar de fusionarlas.

Ya tendríamos, las fuentes de datos clonadas y mezcladas. Ahora faltaría el propio proceso de almacenaje de esta fuente en un archivo. Para ello tendremos que servirnos de un *DataSink*. Recordamos, que un objeto de tipo *DataSink* se usa para leer datos de una *DataSource* y volcarlos en un destino concreto que no sea un dispositivo de presentación. Normalmente se usa para escribir datos en un archivo o a través de una red. Más adelante, en la implementación de “Teletrófono”, escogeremos un formato concreto en el que almacenar este vídeo. Lo deberemos hacer teniendo en cuenta las muchas o pocas, ya veremos, posibilidades que nos ofrece JMF. Escogeremos aquellos códecs y el formato que más nos interese, de acuerdo a los principios y objetivos del programa de videoconferencia que se citan en otros apartados del documento. Para que el archivo de vídeo tenga el formato deseado deberemos pasar la fuente de datos fusionada, por un *Processor* antes de entregársela al *DataSink*. La función de este procesador será pasar la fuente de datos al formato requerido, para que el *DataSink* ya sólo tenga que encargarse de almacenar los datos que salen del *Processor*.

El último de los requisitos en este lado de la comunicación que listábamos unas líneas más arriba, nos recordaba el deseo de que en este lado se puedan ir introduciendo subtítulos. El profesor en el aula debería poder insertar rótulos bajo la imagen recibida desde remoto, por tanto el teclado pasará a ser otro dispositivo activo en el lado local de la videoconferencia. Los subtítulos que se vayan insertando, deberán ir apareciendo por pantalla así como quedarán almacenados en un archivo en este lado de la comunicación. Para llevar a cabo esta funcionalidad no es necesario hacer un diseño a nivel de JMF,

pues no estamos tratando con datos multimedia ni transfiriendo datos a través de la red. Esta funcionalidad, se podrá resolver por tanto, con las clases básicas de Java como *File*, *BufferedWriter*, *FileWriter*, etc.

Finalmente, el diseño de la aplicación de videoconferencia en el lado local de la comunicación, donde se sitúa el profesor con los alumnos, quedaría de la siguiente manera siguiendo el camino detallado en los párrafos anteriores:

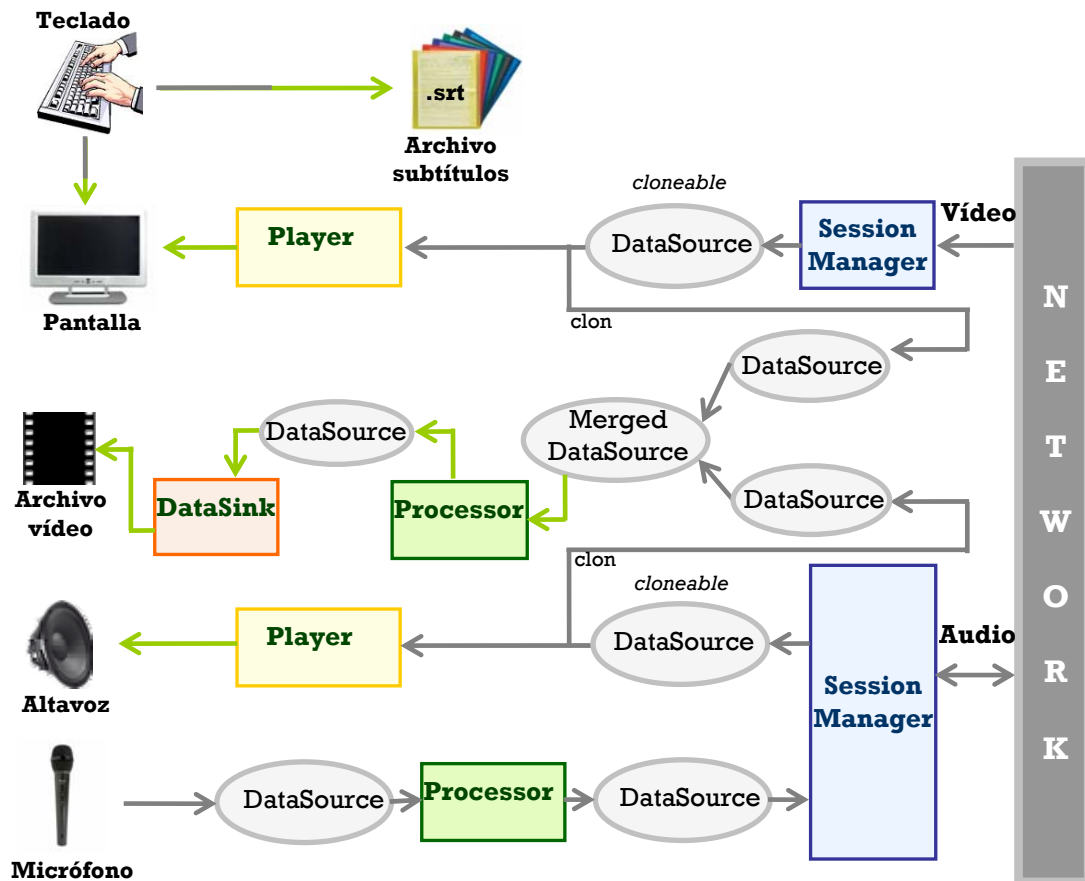


Figura 4.16. Diseño de la aplicación "Teletrófono". Esquema 'C' lado local.

Como se puede observar en el anterior esquema, tanto el audio como el vídeo que recibimos lo estamos convirtiendo en una fuente "clonable". Estas fuentes son las que estamos reproduciendo directamente en tiempo real en el altavoz y la pantalla del ordenador, respectivamente. De las fuentes "clonables" obtenemos un clon para el audio y otro para el vídeo, que mezclaremos entre sí para almacenarlos en el archivo de vídeo.

Pero, como se explicará detalladamente en el apartado 5.3.3 de este proyecto, el anterior esquema nos va a un par de problemas graves cuando tratemos de implementarlo.

### Problema 1:

La máquina virtual de Java dará un error grave que hará que se cierre la aplicación cada vez que intentemos poner en marcha este esquema o por el contrario realizará correctamente sólo una de las labores encomendadas: o bien realiza correctamente la grabación del archivo de vídeo, o bien la reproducción del audio y vídeo, pero nunca realiza correctamente todas las funciones a la vez. Se trató de solucionar el problema modificando levemente el anterior diseño, por ejemplo: utilizando los clones para la reproducción y las fuentes "clonables" para mezclarlas y guardarlas

en el archivo de vídeo. Ninguna de las múltiples modificaciones leves del esquema anterior dieron resultado. Tras consultar posibles soluciones en foros de Internet especializados en JMF, lo único que encontramos fue multitud de personas con este mismo problema.

Finalmente la solución adoptada pasó por rediseñar en parte la totalidad de la aplicación, tanto en el lado remoto como en el lado local de la comunicación. Hemos de saber que una sesión RTP la definen una dirección IP origen, un puerto de origen, una dirección IP destino y un puerto de destino. Si nos fijamos en los dos esquemas de diseño finales de cada lado, para llevar a cabo la aplicación de videoconferencia estamos haciendo uso de únicamente dos sesiones RTP. Es decir, para llevar a cabo la comunicación entre los extremos estamos estableciendo una sesión RTP para enviar el vídeo del lado remoto al local, y otra sesión RTP para establecer una comunicación bidireccional de audio. En total usamos dos puertos en cada extremo. Veámoslo gráficamente de un modo sencillo:

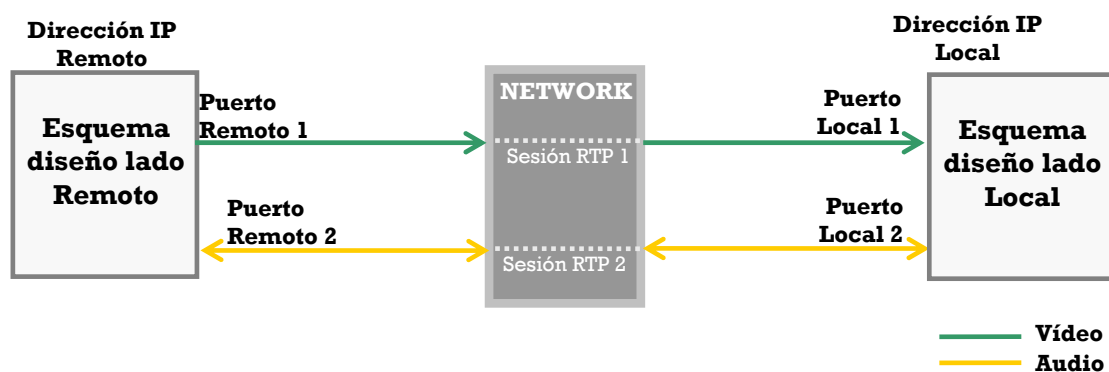


Figura 4.17. “Teletrófono”: Esquema de puertos inicial.

Como vemos, hasta el momento estaba previsto que se estableciesen dos sesiones RTP:

- Sesión RTP 1: Envío de vídeo de (Dirección IP Remoto, Puerto Remoto1) a (Dirección IP Local, Puerto Local 1).
- Sesión RTP 2: Envío y recepción de audio de (Dirección IP Remoto, Puerto Remoto 2) a (Dirección IP Local, Puerto Local 2).

Pero tal y como se ha mencionado, podemos adelantar que este esquema nos va a dar problemas a la hora de implementarlo. El problema reside en el lado local, a la hora de tratar de recibir un único flujo de audio que reproduciremos y clonaremos para su almacenamiento, y a la hora de tratar de recibir un único flujo de vídeo que reproduciremos y clonaremos para su almacenamiento. Tras seguir un proceso de búsqueda de remedios a este problema, que se describe en posteriores apartados de este capítulo, se ha determinado una nueva solución. La nueva solución dará lugar a una aplicación más robusta que no da opción a que la máquina virtual de Java se sature.

La nueva solución propone enviar desde el extremo remoto hasta el local los flujos de audio y vídeo por duplicado, evitando así tener que clonar éstos en el lado local. Es decir, antes enviábamos desde remoto un flujo de audio y un flujo de vídeo, recogíamos ambos flujos en local, clonábamos cada uno de ellos y utilizábamos una pareja audio-vídeo para reproducirla y otra pareja audio-vídeo para llevarla a un

archivo. Ahora enviaremos desde remoto dos flujos de audio y dos flujos de vídeo, recogeremos los cuatro flujos en local y cogeremos, de nuevo, una pareja audio-vídeo para reproducirla y otra pareja para guardarla. Evitaremos de este modo tener que crear fuentes de datos “clonables”, realizar clones y tener que mezclar fuentes clonadas.

Con este nuevo planteamiento solucionamos los problemas en el lado local, pero consumimos más recursos de la red al duplicarse el volumen de información que enviamos a través de la red. Ahora estableceremos cuatro sesiones RTP en vez de dos. Desde luego, no es la solución más eficiente pero sí la más eficaz y segura.

El esquema, por tanto de sesiones RTP que se establecerán finalmente durante una videoconferencia llevada a cabo a través de “Teletrófono” será el siguiente:

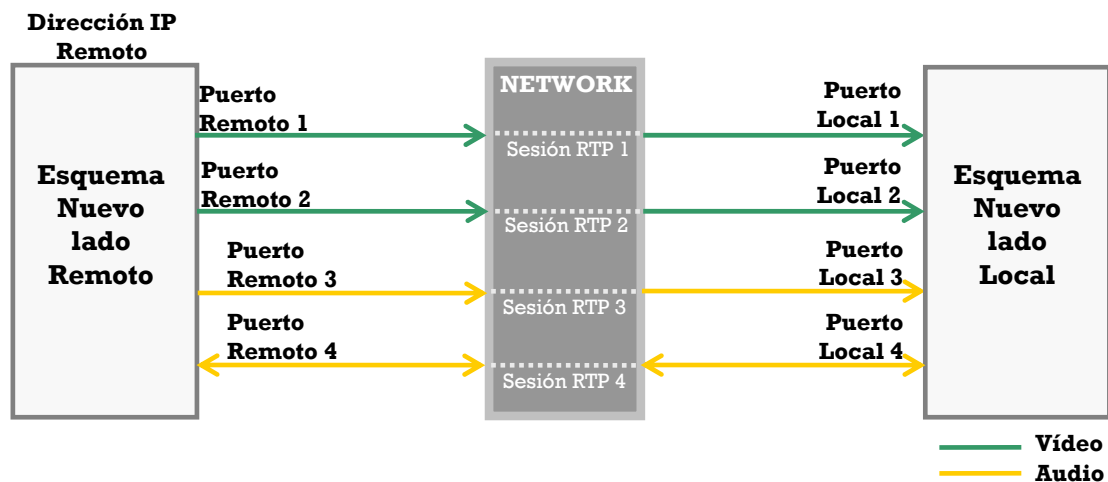


Figura 4.18. “Teletrófono”: Esquema de puertos definitivo.

Las nuevas sesiones RTP quedarán ahora definidas así:

- Sesión RTP 1: Envío de vídeo de (Dirección IP Remoto, Puerto Remoto1) a (Dirección IP Local, Puerto Local 1).
- Sesión RTP 2: Envío de vídeo de (Dirección IP Remoto, Puerto Remoto 2) a (Dirección IP Local, Puerto Local 2).
- Sesión RTP 3: Envío de audio de (Dirección IP Remoto, Puerto Remoto3) a (Dirección IP Local, Puerto Local 3).
- Sesión RTP 4: Envío y recepción de audio de (Dirección IP Remoto, Puerto Remoto 4) a (Dirección IP Local, Puerto Local 4).

Como se ha señalado en la figura, los esquemas de ambos lados de la transmisión deberán variar para adaptarse a estos cambios. Ahora desde el lado remoto debemos mandar la captura de la webcam por dos puertos distintos. Si recordamos, en la figura 4.13 se mostraba como cogíamos los datos de la cámara web y los convertíamos en una fuente “clonable” que mostrábamos por pantalla. A la vez, obteníamos un clon de dicha fuente de datos para enviarlos al otro extremo. Ahora entonces, necesitaremos obtener otro clon de los datos capturados por la cámara, para enviarlos también al otro extremo.

Algo parecido sucederá con los datos capturados por el micrófono en el lado remoto. Antes simplemente cogíamos estos datos y los transmitíamos por la red. Ahora habrá que convertirlos en una fuente “clonable”, que se seguirá enviando a la red, y obtener de ella un clon que también enviaremos al otro extremo a través de la red.

En el lado local también variará el diseño. Ahora recibimos en ese lado cuatro flujos RTP. El diseño quedará simplificado en tanto en cuanto ya no necesitamos clonar fuentes de datos a este lado de la transmisión. Tan sólo cogeremos uno de los flujos de audio que nos llegan y lo reproduciremos, cogeremos uno de los flujos de vídeo que nos llegan y lo reproduciremos, y los otros dos flujos restantes de audio-vídeo serán los que almacenemos.

## Problema 2:

Como se explicará al detalle en el apartado 4.2.3, si seguimos el esquema de diseño que teníamos configurado hasta ahora nos surge un problema con el archivo de vídeo en el que hemos almacenado la imagen y el audio que llega desde el lado remoto. En concreto lo que ocurre es que obtenemos una falta de sincronización entre las dos pistas que conforman el archivo. Tras un proceso de investigación, que se explica detenidamente más adelante, se llegó a la conclusión de que la mejor solución era almacenar la pista de audio en un archivo y la pista de vídeo en otro archivo distinto. Esto conllevó una nueva modificación del diseño de la aplicación. Ahora prescindimos entonces de la *MergedDataSource* y lo que hacemos es pasar directamente cada una de las *DataSources* recibidas a dos *Processors* distintos que adecúan sus respectivos formatos; después cogemos sendas *DataSources* de salida y las pasamos a dos *DataSinks* distintos. Uno de los *DataSinks* se encarga de almacenar el flujo de vídeo en un archivo y el otro hace lo mismo pero con el flujo de audio.

Todos estos cambios se aprecian claramente en el esquema que se muestra en las Figuras 4.19 y 4.20. Estas figuras muestran el diseño definitivo de la porción de “Teletrófono” que implementa el servicio de videoconferencia educativa.

## EXTREMO REMOTO DE LA VIDEOCONFERENCIA

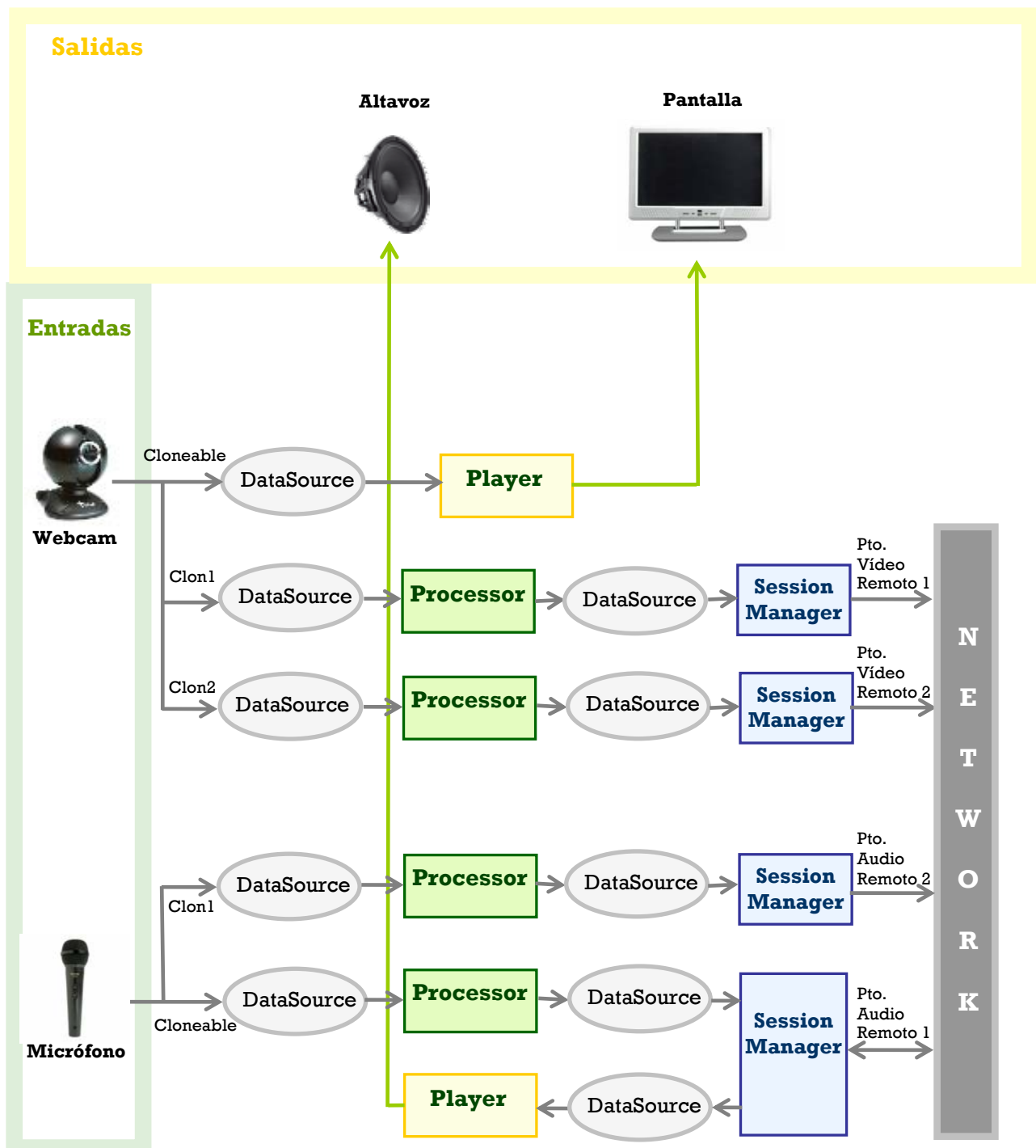


Figura 4.19. “Teletrófono”: Esquema definitivo Extremo Remoto.

## EXTREMO LOCAL DE LA VIDEOCONFERENCIA

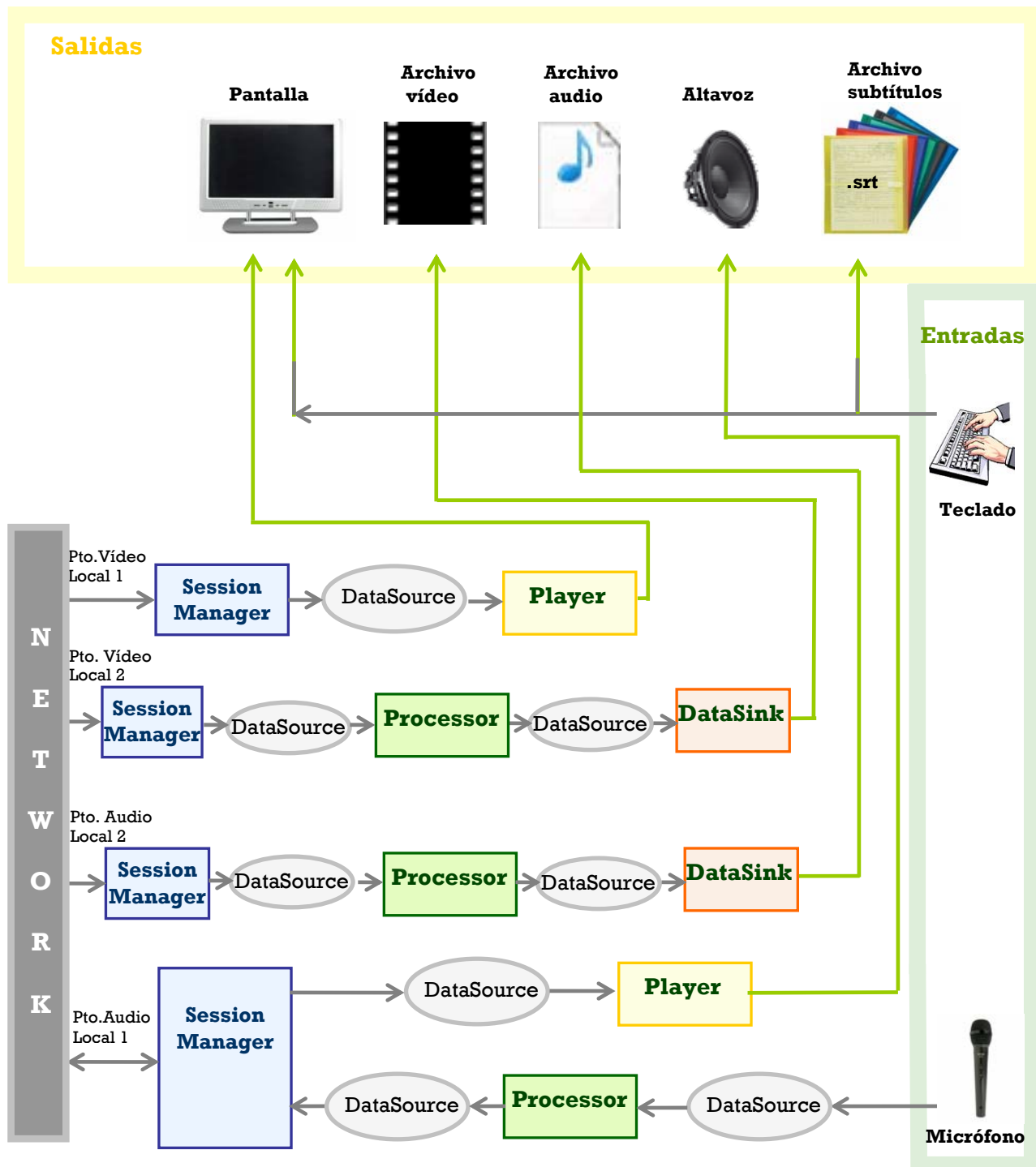


Figura 4.20. “Teletrófono”: Esquema definitivo Extremo Local.



Recordemos que “Teletrófono” nos ofrece dos herramientas más aparte de la herramienta principal de establecimiento de videoconferencia en la que nos hemos venido centrando todo este apartado. Dichas herramientas son el reproductor y el editor de subtítulos.

El reproductor de “Teletrófono” está especialmente pensado para reproducir las grabaciones de las sesiones de videoconferencia. Al iniciar el reproductor, éste nos deberá pedir el archivo que queremos reproducir. Una vez el usuario haya proporcionado dicho archivo, el reproductor comprobará si éste tiene un archivo de audio (.gsm) o de subtítulos (.srt) asociado. Las posibilidades son varias suponiendo que el archivo de vídeo indicado por el usuario sea de un tipo válido (‘.avi’, ‘.mov’ o ‘.mpg’):

- que el archivo sea de tipo ‘.mov’ o ‘.mpg’ (con o sin subtítulos asociados) → en cuyo caso se reproducirá únicamente el archivo de vídeo (y sus subtítulos en caso de que existan);
- que el archivo sea de tipo ‘.avi’ sin archivo de audio asociado (con o sin subtítulos asociados) → en cuyo caso se reproducirá únicamente el archivo de vídeo (y sus subtítulos en caso de que existan);
- que el archivo sea de tipo ‘.avi’ con archivo de audio asociado (con o sin subtítulos asociados) → en cuyo caso se reproducirán el archivo de vídeo y el archivo de audio simultáneamente (y sus subtítulos en caso de que existan).

Para implementar este reproductor nos serviremos de las clases básicas de Java como *JFileChooser*, *JFile*, *JFrame*, *JButton*, etc. Pero también deberemos acudir al paquete JMF para poder reproducir el archivo de vídeo que el usuario escoja.

En el caso en que sólo haya que reproducir un archivo multimedia, deberemos construir un *MediaLocator* que identifique la ruta donde se encuentra el archivo escogido por el usuario y, a partir de éste, crear el *Player*. Para que el usuario pueda visualizar por pantalla el vídeo, deberemos obtener la componente visual de dicho *Player* y añadirla a nuestro panel de reproducción. También podemos añadir a nuestro panel los controles del *Player* que creamos oportunos.

En el caso en el que la grabación haya sido creada con Teletrófono deberemos reproducir dos archivos multimedia (audio y vídeo) simultáneamente. En ese caso habrá que crear dos *MediaLocators* y dos *Players*, uno para cada pista.

En la figura 4.21 se representa el esquema que habría que seguir en el caso en que quisiésemos reproducir una grabación realizada con Teletrófono en la que se hubiesen introducido subtítulos.

La herramienta de edición y re-sincronización de subtítulos funcionará de modo muy similar al reproductor. De hecho, esta herramienta incluye dentro un reproductor de vídeo. Nos pedirá por tanto también, la ruta del archivo de vídeo que nos interesa, en este caso, editar y comprobará que exista un archivo de audio u otro de subtítulos asociados a este vídeo. Con esta herramienta aparte de visualizar el vídeo escogido, podremos modificar el contenido de los subtítulos asociados y los tiempos de inicio y fin de cada uno de ellos. Para implementar esta herramienta veremos más adelante que las clases más usadas han sido aquellas que pertenecen al paquete básico de Java, por ejemplo: *File*, *BufferedReader*, *FileReader*, *BufferedWriter*, *FileWriter*, etc. Esto se debe a que la mayoría de operaciones que se realizan desde esta herramienta de

“Teletrófono”, son para leer, escribir o modificar el archivo que contiene los subtítulos asociados al vídeo escogido. Pero, como hemos dicho, esta herramienta engloba también un reproductor, por tanto deberemos hacer uso del paquete JMF exactamente del mismo modo que hemos explicado para el reproductor de “Teletrófono” en el párrafo anterior y a través de la figura 4.21.

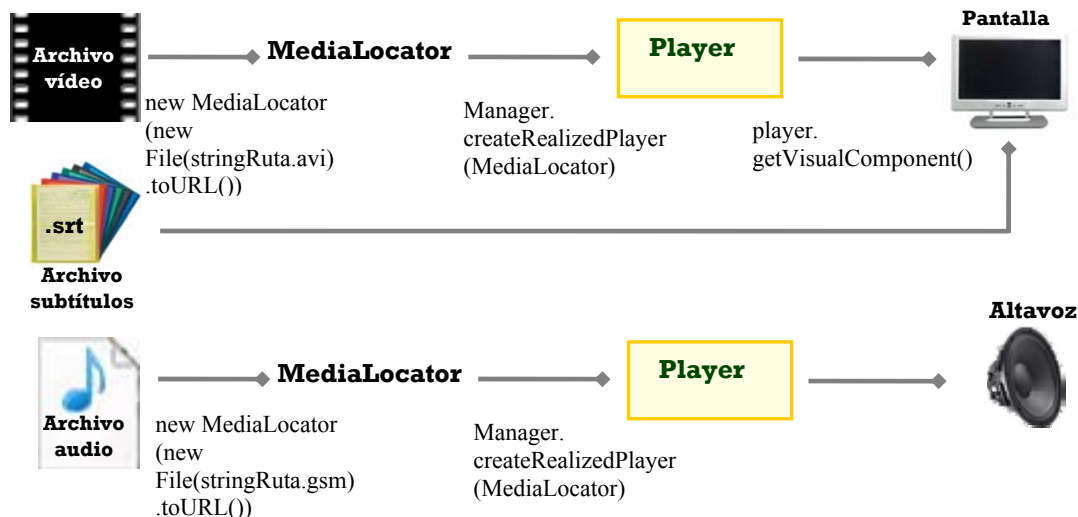


Figura 4.21. Reproductor de “Teletrófono”.

### 4.2.3. IMPLEMENTACIÓN

En este apartado trataremos de implementar el esquema definitivo de videoconferencia que hemos diseñado para nuestra aplicación, así como las herramientas de reproducción y edición y re-sincronización de subtítulos que Teletrófono incluye. Se irá narrando el proceso seguido así como se mencionarán algunos de los problemas que han ido surgiendo y como se han ido solucionando.

La clase principal de nuestra aplicación que contendrá el método *main* y lanzará el resto de la aplicación, será la clase ‘**Presentación**’. Esta clase hereda de la clase ‘*JFrame*’ de Java para mostrar una ventana de presentación del programa en la que aparece el nombre de éste y el nombre de sus creadores. Queríamos que esta ventana de presentación apareciese durante siete segundos, para después ocultarse y dar paso al menú principal de la aplicación. Para lograr esto, hubo que crear un objeto de la clase ‘*Timer*’, del paquete *swing* de Java, y asociarle un determinado oyente. Cuando pasan esos siete segundos desde que se inició el *timer*, la aplicación salta al código del oyente, en el que se indica que se debe cerrar esta ventana y lanzar la siguiente clase de la aplicación: ‘**MenúPrincipal**’.



Figura 4.22. Pantalla de presentación de Teletrófono.

La clase '**MenúPrincipal**', como su propio nombre indica, será la ventana de la aplicación en la que se muestren las diferentes opciones que nos ofrece el programa. Esta clase también hereda de la clase '*JFrame*' y se le ha añadido una barra de menú (*JMenuBar*). Los menús (*JMenu*) disponibles son 'Programa', que a su vez alberga el *JMenuItem* 'Salir', y 'Ayuda', que alberga a los *JMenuItem* 'Acerca de...' y 'Manual de Ayuda'. El elemento 'Salir' simplemente pregunta al usuario si está seguro de querer cerrar la aplicación y en caso de respuesta afirmativa, se cerrará la ventana del menú y se finalizará la ejecución del programa. El elemento 'Acerca de...' lanzará la clase '**DiálogoAcercaDe**', que hereda de la clase '*JDialog*' y muestra un diálogo con los autores y fechas de creación de la aplicación. Por último el elemento 'Manual de Ayuda' lanzará la clase '**DiálogoAyuda**', que hereda de '*JDialog*' y muestra un *JTextArea* con un texto que ayudará al usuario a saber utilizar el programa.

En la ventana de menú principal aparecen cuatro imágenes que hacen referencia a cada una de las cuatro acciones que podemos llevar a cabo. El usuario deberá pulsar sobre aquella imagen que haga referencia a la tarea que desea realizar. Cuando el usuario se sitúa sobre una de estas cuatro imágenes, dicha imagen desaparece y aparece en su lugar el nombre de la herramienta sobre la que está situado. Veámoslo con un ejemplo. A continuación aparece la imagen inicial de la ventana de menú principal:

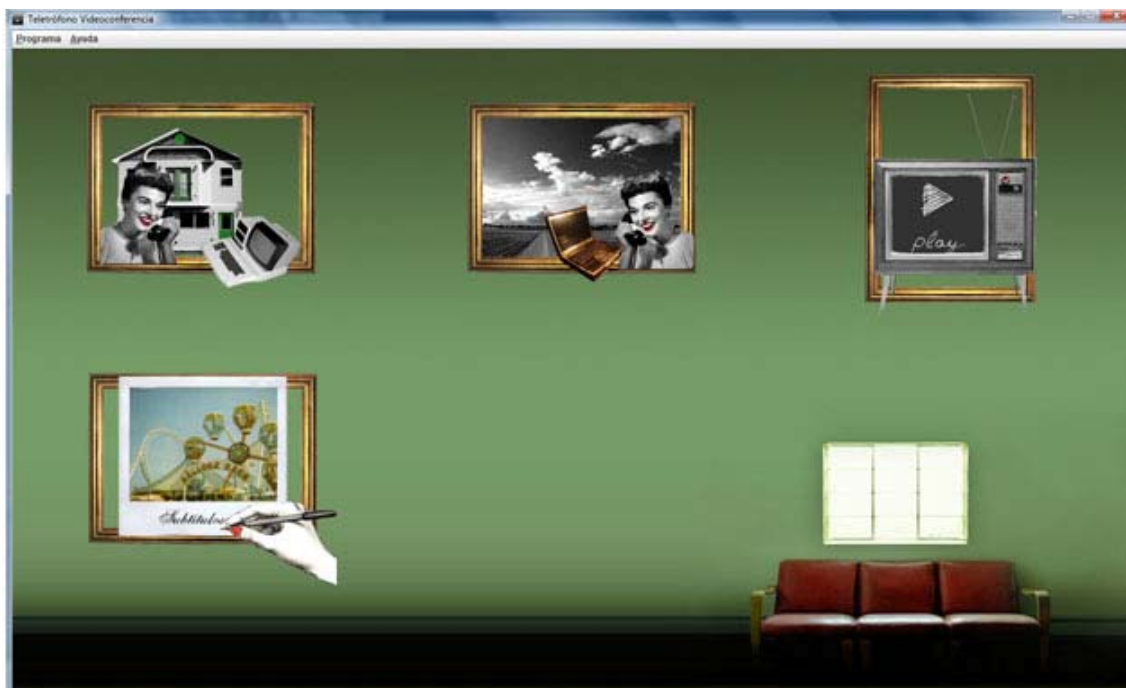


Figura 4.23. Ventana del menú principal de Teletrófono.

Observamos que hay cuatro “cuadros” colgados en esta sala de espera:

- En el primer icono aparece una chica en su casa realizando una videoconferencia; esta será la opción que debamos escoger para iniciar una videoconferencia desde el extremo local.
- En el segundo icono cuadro aparece una chica, también realizando una videoconferencia, pero esta vez desde un lugar al aire libre; esta será la opción que debamos escoger para realizar una videoconferencia desde el extremo remoto.
- En el tercer icono aparece una televisión con el símbolo de ‘Play’; será el icono que debamos escoger para iniciar el reproductor.
- En el último icono aparece una imagen con un subtítulo que está siendo editado; será la opción que debamos elegir para editar y re-sincronizar los subtítulos de un vídeo.

Pero para que los iconos, en realidad botones, no resulten confusos, cada vez que el usuario sitúa el cursor encima de alguno de ellos, aparece una frase que explica la funcionalidad de éste. En la figura 4.24, podemos ver lo que pasa si, por ejemplo, nos situamos sobre el icono para iniciar una videoconferencia desde el extremo remoto.

Como vemos, cuando nos posicionamos sobre uno de los cuatro iconos aparece una frase explicativa. En este caso, aunque no se aprecia bien por las dimensiones de la imagen, cuando nos posicionamos sobre el icono central de la ventana, aparece la frase: “Iniciar Videoconferencia desde el extremo Remoto”. Al posicionar el cursor sobre los otros iconos aparecerán las frases: “Iniciar Videoconferencia desde el extremo Local”, “Reproducir grabación” y “Editar y Re-sincronizar Subtítulos”.



Figura 4.24. Menú principal de Teletrófono: “Iniciar Videoconferencia desde el extremo Remoto”.

Como hemos dicho, cada uno de los cuatro iconos son en realidad botones. La ventana de menú principal contiene, por tanto, cuatro objetos de una clase propia que hemos llamado **‘Botón’**. La clase **‘Botón’** hereda de la clase **‘JButton’** de Java y define un nuevo constructor que recibe como parámetros tres objetos *Icon* que hacen referencia a tres imágenes. El constructor crea un objeto *JButton* con una de las imágenes, pero elimino su borde típico cuadrado, dejo su área transparente y llamo a los métodos de *JButton* *setRolloverIcon* y *setPressedIcon*, pasándoles como parámetros las dos imágenes restantes.

Hay otra clase de la aplicación que debemos nombrar, ya que también se puede lanzar desde la ventana del menú. Se trata de la clase **‘DiálogoSalir’**. Cuando estamos en el menú principal de la aplicación y pinchamos el símbolo ‘X’ que aparece en la esquina superior derecha de la ventana, se lanzará la clase mencionada. También se lanzará esta clase si tratamos salir del programa a través del *MenuItem* “Salir”. Dicha clase hereda de *JDialog* y muestra un diálogo en el que se pregunta al usuario si está seguro de que desea salir del programa. Aparecen tras la pregunta dos botones: “Aceptar” y “Cancelar”. Si aceptamos, se cerrará la aplicación por completo. Si cancelamos, tan sólo se cerrará este diálogo. En todas las ventanas de Teletrófono que el botón de la ‘X’ (situado en la esquina superior derecha de éstas) esté habilitado, se lanzará la clase **‘DiálogoSalir’**. Es una forma de que los usuarios del programa no cierren la aplicación por equivocación.

Una vez que hemos revisado las clases que lanza la aplicación en su inicio, vamos a centrarnos en cada uno de los elementos del menú.



#### 4.2.3.1.- El Reproductor Teletrófono

Comencemos por la herramienta más sencilla: el Reproductor Teletrófono. Cuando el usuario escoge la opción “Reproducir grabación” en el menú principal, se lanza automáticamente la clase ‘**ChooserAbrir**’. Dicha clase se encarga de mostrar al usuario un *JFileChooser*, es decir, una ventana a través de la cual puede navegar por los directorios de su ordenador y escoger aquel archivo que desee. Sin embargo, al constructor de esta clase se le debe indicar qué tipo de archivos mostrará por defecto el selector. En este caso el selector deberá llamar a la clase ‘**FiltroVideo**’, que hace que el selector sólo muestre archivos de vídeo cuya extensión sea ‘.mov’, ‘.avi’ o ‘.mpg’. Una vez el usuario haya escogido un archivo y pulse el botón “Aceptar”, se comprobará que sea un archivo válido y se ejecutará finalmente la clase ‘**Reproductor**’. Consideraremos un archivo como válido siempre que exista, que se pueda leer y que su extensión sea ‘.mov’, ‘.avi’ o ‘.mpg’. En caso de que no cumpla alguna de estas condiciones se mostrará un mensaje informativo al usuario, se volverá a la pantalla del menú principal y no se mostrará el reproductor.

Antes de seguir describiendo la implementación del Reproductor es importante saber que las grabaciones de las sesiones de videoconferencia que llevemos a cabo mediante Teletrófono quedarán almacenadas (tal y como explicaremos detalladamente en futuros apartados) en un archivo ‘.avi’, que contendrá el vídeo, un archivo ‘.gsm’, que contendrá el audio y, si se han introducido subtítulos, se creará también un archivo ‘.srt’. Por tanto, aunque nuestro reproductor está preparado para reproducir archivos con o sin subtítulos, cuya extensión sea ‘.avi’, ‘.mov’ o ‘.mpeg’, deberemos crear un tipo de reproductor u otro dependiendo de si el archivo es una grabación de sesión realizada con Teletrófono o no. El esquema a seguir sería:

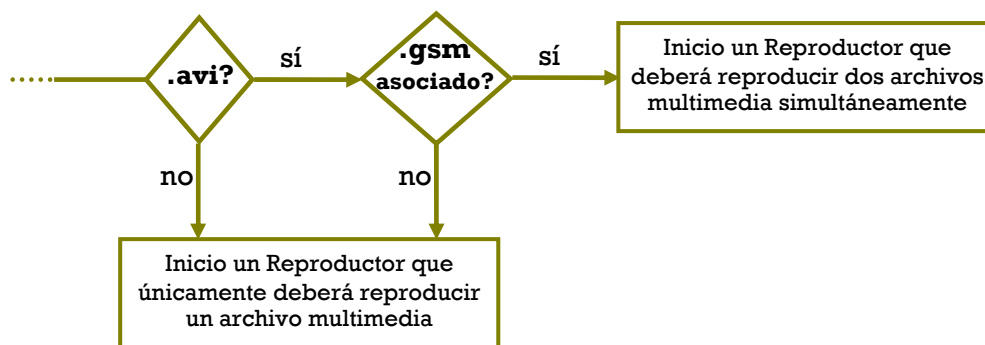


Figura 4.25. Esquema implementación de parte del Reproductor Teletrófono.

Como vemos en la anterior figura, lo primero que hay que comprobar es si el archivo escogido por el usuario es un archivo AVI (*Audio Video Interleave*); en caso de que no lo sea estaremos sin duda ante un archivo de vídeo que no se ha grabado con Teletrófono. Si por el contrario el archivo escogido por el usuario es un AVI, pasaremos a comprobar si dicho archivo tiene un archivo con la extensión ‘.gsm’ asociado. Consideramos que tiene un archivo GSM asociado siempre y cuando exista un archivo ‘.gsm’ en la misma ruta y con el mismo nombre (a excepción de la extensión) que el AVI especificado por el usuario. Si no hay ningún archivo asociado de este tipo estaremos de nuevo ante un vídeo que no ha sido creado a través de Teletrófono. Si por el contrario encontramos un archivo GSM asociado, estaremos ante una grabación de

una sesión de videoconferencia Teletrófono. Dependiendo de si hemos concluido que es una grabación Teletrófono o no, llamaremos al constructor de la clase Teletrófono de una u otra forma.

La clase “Reproductor” hereda de la clase *JFrame*, por lo que será una ventana. El constructor de la clase, habrá recibido como parámetros una cadena de texto con la ruta en la que se encuentra el archivo de vídeo a reproducir y una variable *booleana* que será verdadera si se concluyó que el archivo escogido por el usuario fue grabado con Teletrófono (es un ‘.avi’ con un ‘.gsm’ asociado), y será falsa si se concluyó que el archivo que el usuario eligió no se creó con Teletrófono. Lo primero que se hará, tanto si es una grabación Teletrófono como si no, será comprobar si existe algún archivo de subtítulos asociado al archivo de vídeo escogido. Para que se considere un archivo de subtítulos asociado, deberá estar ubicado en el mismo directorio que el archivo de vídeo y llamarse igual, a excepción de la extensión del archivo. Por ejemplo, si lanzo el reproductor y en el selector de archivos selecciono un archivo que hay en mi escritorio que se llama “video\_prueba.mov”, mi reproductor buscará si existe algún archivo en mi escritorio que se llame “video\_prueba.srt”. Los archivos de subtítulos siempre serán archivos con la extensión ‘.srt’. En el caso en que no exista un archivo de subtítulos asociado, no hay problema, el reproductor mostrará por pantalla únicamente el archivo de vídeo escogido. Para crear un *Player* que reproduzca el archivo situado en la ruta que el usuario indicó utilizaremos la expresión:

```
Manager.createRealizedPlayer(new MediaLocator(new File(rutaArchivo).toURL()));
```

Como vemos, se ha de crear un *MediaLocator* que haga referencia a la ubicación del archivo. Después pasamos ese localizador al método del *Manager* *createRealizedPlayer*. Podíamos haber usado el método tradicional *createPlayer*, sin embargo es más cómodo usar *createRealizedPlayer*. Este último es un método de bloqueo que, aparte de crear el *Player*, llama al método *realize* de éste y retorna cuando finalmente el *Player* ha pasado a estado *Realized*. Pasar un *Player* a estado *Realized* puede ser una operación que consume mucho tiempo, se debe tener precaución al usar este método pues puede bloquear el curso del programa durante varios segundos.

En el caso en que el archivo a reproducir haya sido creado con Teletrófono deberemos crear dos *Players* uno para el archivo de vídeo ‘.avi’ y otro para el archivo de audio ‘.gsm’.

Tras crear el *Player* o *Players* necesarios, bastará con ejecutar la sentencia *playerVideo.getVisualComponent()* para obtener el componente visual del reproductor que se encarga de reproducir el vídeo (no el audio). Ahora bastaría con añadir dicho componente visual al panel deseado de nuestro *JFrame* “Reproductor”. Para comenzar la reproducción, en el caso en que no estemos ante una grabación Teletrófono, pondremos *playerVideo.start()*.

Sin embargo, en el caso de que estemos ante una grabación Teletrófono deberemos lanzar al mismo tiempo el reproductor del audio y el reproductor de vídeo de modo que ambos estén sincronizados. ¿Cómo hacemos esto?, pues consultando en el manual de JMF, podemos leer que tenemos varias opciones, pero recomiendan una en concreto: teniendo en cuenta que un *Player* puede usarse para dirigir las operaciones de un *Controller*, podemos usar uno de los *Players* para dirigir al otro. Un *Player* pasaría a ser el “*managing Player*” y el otro el “*managed Player*”. El “*managing Player*” se ocupará entonces de la gestión de los estados y de la sincronización con el “*managed Player*”. Este mecanismo se implementa a través de los métodos *addController* y *removeController*. Para poder realizar esta operación los *Players* deben estar en el

estado *Realized*. Una vez que hemos añadido uno de los *Players* como *Controller* del otro *Player*, deberemos interactuar únicamente con el ‘*managing Player*’ y automáticamente los métodos se propagan al ‘*managed Player*’. Entonces, en nuestro programa Teletrófono, cuando se trata de una grabación Teletrófono hacemos:

```
try{ player.addController(playerAudio);
}catch(IncompatibleTimeBaseException e){ e.printStackTrace();}
player.setRate(player.getRate());
player.start();
```

; con la primera sentencia hemos añadido el *Player* de audio como *Controller* al *Player* de vídeo, con lo cual a partir de ahora sólo deberemos interactuar con el *Player* de vídeo. A continuación con la siguiente sentencia nos aseguramos que la velocidad de reproducción de los fotogramas del vídeo sea la misma. Con la última sentencia comenzamos los dos *Players* de forma síncrona, a pesar de estar ejecutando la sentencia *start* tan sólo en el *Player* de vídeo.

En el caso en que el archivo de vídeo escogido tenga subtítulos asociados, la complejidad del código aumenta, pues aparte de poner en marcha el *Player* (o los *Players*), hay que mostrar dichos subtítulos bajo la imagen del vídeo. Estos subtítulos irán cambiando conforme vaya avanzando la reproducción, de acuerdo con los tiempos que aparezcan en el archivo que los contiene. Los archivos de subtítulos serán archivos con la extensión ‘.srt’ y cuyo formato deberá ser el siguiente:

- En la primera línea de un subtítulo deberá aparecer un número, que indica en qué número de subtítulo estamos. La numeración empezará siempre en ‘1’ para cada archivo de subtítulos y se seguirá una numeración ascendente correlativa de uno en uno: ‘1’, ‘2’, ‘3’, etc.
- La segunda línea correspondiente a un subtítulo contendrá los tiempos de inicio y de fin del subtítulo. Los tiempos seguirán el formato ‘hh:mm:ss,kkk’, donde ‘hh’ es la hora, ‘mm’ son los minutos, ‘ss’ son los segundos, y ‘kkk’ son los milisegundos. El formato de la línea será:  
`hh:mm:ss,kkk --> hh:mm:ss,kkk`  
 donde el primer tiempo que aparece es el tiempo de inicio del subtítulo y el segundo tiempo que aparece es el tiempo de finalización del subtítulo. El tiempo de inicio de un subtítulo siempre deberá ser mayor que el tiempo de finalización del subtítulo anterior. Es decir, los subtítulos deberán aparecer en el documento siguiendo el orden de aparición y los tiempos no deberán solaparse.
- La tercera línea correspondiente a un subtítulo contiene el propio texto de éste. Si el texto del subtítulo es largo, ocupará las líneas sucesivas. La longitud máxima de un subtítulo es de 125 caracteres.
- La última línea correspondiente a un subtítulo deberá ser una línea en blanco, que actuará como indicador de final de subtítulo.

Veamos en la figura 4.26 un ejemplo de archivo de subtítulos que contiene tres subtítulos:



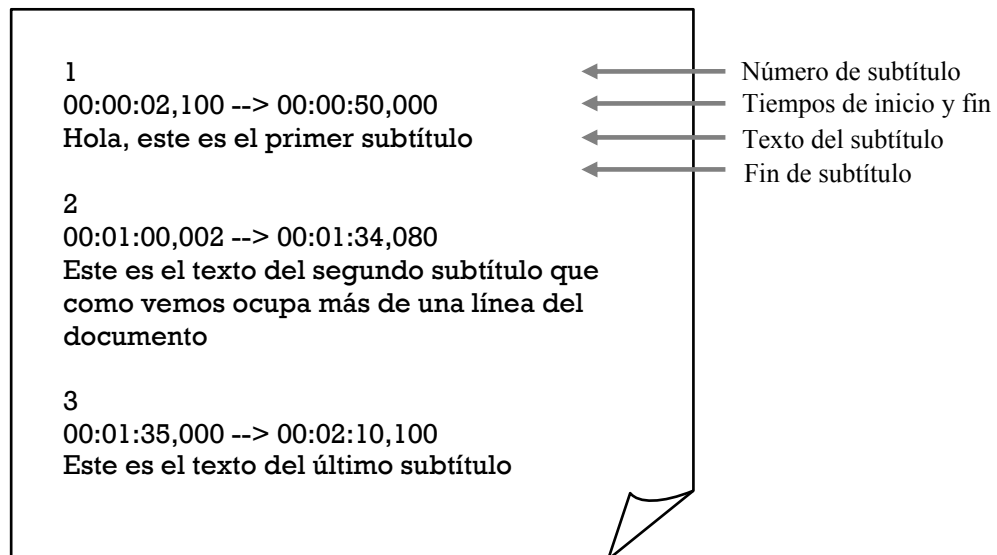


Figura 4.26. Formato del archivo SRT con los subtítulos.

Este es el tipo de archivo de subtítulos que Teletrófono crea cuando se graba una videoconferencia en la que se han ido introduciendo rótulos desde el extremo local. Es por ello que éste es el formato de archivo de subtítulos que el reproductor de Teletrófono espera recibir. Se pueden introducir un máximo de cincuenta subtítulos y cada subtítulo podrá tener una longitud máxima de ciento veinticinco caracteres. Por lo que esas serán las características de los subtítulos que el reproductor espera.

Este tipo de archivos de texto están pensados para que se creen, se modifiquen y se reproduzcan mediante Teletrófono, a través de sus herramientas de videoconferencia, el editor y el reproductor, respectivamente. Es por ello aconsejable, que se eviten modificaciones de estos archivos fuera de Teletrófono ya que podrían dar lugar a errores cuando quisiésemos volver a usarlos con nuestro programa.

El Reproductor Teletrófono no comprueba que el archivo de subtítulos asociado al vídeo que se desea reproducir siga este formato. Sin embargo, si introducimos un archivo que no siga dicho formato, la propia aplicación hará que salte una o varias excepciones y al usuario le aparecerá por pantalla un mensaje de error en el que se le indicará que no se ha podido crear un reproductor para el archivo seleccionado.

En el caso en que el archivo de subtítulos sea modificado manualmente y no a través de Teletrófono, acción que se desaconseja, veremos qué ocurre en los siguientes casos en los que, aunque el formato es el adecuado, los datos que contiene no lo son:

- Imaginemos que los números de los subtítulos no empiezan en '1', o no son números correlativos, están desordenados, o ni siquiera es un número sino una letra. En todos estos casos, el reproductor actuará del mismo modo. Se comenzará la reproducción del vídeo y los subtítulos irán apareciendo simplemente en el orden en el que aparecen en el documento.
- Si un usuario modifica o crea manualmente un archivo de subtítulos e introduce en vez de la hora de inicio o fin, letras, el reproductor mostrará un mensaje de error al usuario en el que se indicará que no se ha podido crear un reproductor para el archivo de vídeo seleccionado.

- Si un usuario elimina, o simplemente no introduce, una línea vacía entre subtítulo y subtítulo, el reproductor hará caso omiso y reproducirá de todos modos el archivo de vídeo y sus subtítulos correctamente.
- Imaginemos ahora que los subtítulos no aparecen ordenados por el orden en el que deben reproducirse, sino que el que en el documento aparece como segundo subtítulo tiene tiempos de inicio y fin anteriores al subtítulo que aparece en la primera posición del documento. En este caso la reproducción del vídeo se realizará correctamente, pero la reproducción de los subtítulos no será correcta. No aparecerán los subtítulos en los tiempos que deben o incluso puede que alguno de ellos no aparezca. El usuario debería darse cuenta de este problema y revisar el documento con subtítulos que él mismo habrá creado o modificado manualmente.
- Si los subtítulos que aparecen en el archivo tuviesen definidos tiempos que hiciesen que se solapase un subtítulo con otro, ocurriría lo mismo que en el caso anterior. La reproducción del vídeo se realizará correctamente, pero la reproducción de los subtítulos no será correcta. No aparecerán los subtítulos en los tiempos que deben o incluso puede que alguno de ellos no aparezca.
- Por último consideremos la opción de que un usuario cree manualmente el archivo con los subtítulos e introduzca para alguno de ellos un texto de más de 125 caracteres de longitud. Ante tal caso, el reproductor mostrará correctamente por pantalla el archivo de vídeo y los subtítulos. Sin embargo, de aquellos subtítulos que superen la citada longitud, sólo se mostrarán sus primeros 125 caracteres.

Ya sabemos como funciona el reproductor en el caso en el que no exista un archivo de subtítulos asociado al vídeo, y sabemos como debe ser un archivo de subtítulos. Veamos ahora cómo se ha implementado la reproducción de los rótulos. Suponemos entonces que existe un archivo ‘.srt’, con el formato y características adecuadas, que contiene los subtítulos asociados al vídeo escogido. Los subtítulos deberían ir apareciendo en un área de texto (*JTextArea*) que se ha colocado bajo el componente visual del *Player* del vídeo.

Para comenzar, deberemos implementar un método dentro de la clase “Reproductor” que cree un fichero con la ruta en la que se encuentra el archivo de subtítulos y vaya leyendo los datos que ese fichero contiene. Para ello ejecutaremos la sentencia

```
File fileSubt = new File(rutaArchivoSubtítulos);
```

Comprobaremos que ese fichero existe y se puede leer y crearemos un *FileReader* que nos servirá para crear a su vez un *BufferedReader* del siguiente modo:

```
BufferedReader entrada = new BufferedReader(new FileReader(fileSubt));
```

Luego iremos leyendo el fichero línea a línea, almacenando temporalmente cada línea en una cadena de texto: `String texto = entrada.readLine();`

Por último extraeremos los datos que nos interesen de cada una de esas cadenas para guardarlos en variables locales con forma de *array*. En concreto tendremos un *array* con los tiempos de inicio de todos los subtítulos, otro *array* con los tiempos de finalización de todos los subtítulos y, por último, un *array* que contiene los textos de cada subtítulo. Así habremos finalizado la importación de datos del archivo de subtítulos a nuestra clase ‘Reproductor’.

¿Qué hacemos ahora para reproducir los subtítulos? Pues deberemos crear un objeto de tipo *Timer*, del paquete *swing* de Java, que lleve asociado un oyente concreto y se ponga en marcha cuando se inicia la reproducción del vídeo. El oyente se deberá accionar cada

vez que el tiempo de reproducción del vídeo alcance un tiempo igual a cualquiera de los que tenemos almacenados como tiempos de inicio o de fin de subtítulo. Cuando el oyente salte al oír un tiempo de inicio, deberá mostrar en el área de texto el contenido del subtítulo correspondiente a ese tiempo de inicio. Cuando el oyente salte al oír un tiempo de fin, deberá borrar el contenido que hubiese en el cuadro de texto, dejándolo vacío.

Como ya se ha comentado, este reproductor está especialmente diseñado para reproducir archivos de vídeo que hayan sido grabados a través de la herramienta de grabación de videoconferencias de la que está dotado el programa Teletrófono. Por tanto, está diseñado especialmente para archivos de vídeo que lleven asociado un archivo de subtítulos. En este caso, a la hora de reproducir los subtítulos, no es deseable que se muestren los controles del *Player* de JMF, pues esos controles nos permitirían avanzar o retroceder rápidamente en la reproducción, lo que complicaría sustancialmente la tarea de reproducir los subtítulos de forma síncrona con el vídeo.

Es por ello por lo que se ha evitado mostrar los controles por defecto del *Player*. A cambio, para no perder todo el control del vídeo, se han creado dos controles básicos, uno para parar e iniciar la grabación desde el principio cuando queramos (*Play/Stop*), y otro para poder pausar y reanudar la reproducción del vídeo (*Pause/Resume*). Además, hemos añadido en la parte inferior derecha de la reproducción un reloj que va mostrando el minutaje del vídeo que está reproduciéndose. Para llevar a cabo esta última tarea, hemos creado una clase llamada '**Reloj**'.

La clase "Reloj", hereda de la clase '*JPanel*' y se trata, por tanto, de un panel en el que por medio de iconos que representan los números del 1 al 9, aparece un reloj tipo digital con la hora, minutos y segundos. Se va actualizando cada segundo a través de un *Timer* (del paquete *swing* de Java). Podemos poner en marcha, *resetear*, pausar u obtener o cambiar la hora, los minutos o los segundos en cualquier momento.

Debido a la sencillez y facilidad de uso de la clase 'Reloj', también la hemos utilizado para crear un objeto auxiliar en el reproductor que nos permita pausar la grabación sin que se pierda la sincronización de los subtítulos. En este caso no haremos el objeto visible. El método que hemos seguido para poder pausar la grabación sin perder la sincronización es el siguiente: cada vez que salta el oyente del *Timer* principal porque se ha llegado a un tiempo de inicio de subtítulo, crearemos un objeto Reloj, al que llamaremos '*rAux*', y lo pondremos en marcha. También restaremos el tiempo de fin de subtítulo menos su tiempo de inicio y crearemos un nuevo *Timer* con ese valor. El oyente no saltará entonces hasta que llegue el tiempo de finalización de subtítulo, tiempo durante el cual teóricamente no hay que llevar a cabo ninguna acción. Pero si el usuario pausa la reproducción antes de que llegue ese tiempo de finalización, nos encontramos con un problema, pues el *Timer* no se puede pausar, éste seguirá contando a menos que lo paremos. Por tanto, lo que haremos cuando el usuario pause la reproducción, será parar el *Timer* pero guardar el tiempo exacto con el que se creó, lo que se llama *InitialDelay*. Pararemos también '*rAux*'. Cuando el usuario reanude la reproducción, bastará con crear un nuevo *Timer* cuyo *InitialDelay* sea la resta del *InitialDelay* del anterior *Timer* menos el valor que tenía '*rAux*' cuando se pausó la grabación. Veamos un ejemplo:

- Supongamos que tenemos un subtítulo cuyo tiempo de inicio es el 00:00:12,000 y cuyo tiempo de fin es 00:00:20,000. Si hemos configurado correctamente el *Timer* inicial, en el segundo 12 deberemos estar en el código del oyente del *Timer*. Allí buscaremos el texto del subtítulo asociado a ese tiempo de inicio y lo mostraremos

en el área de texto del reproductor. Se creará un nuevo *Timer* con un *initialDelay* de  $(20,000-12,000) = 8$  segundos. Dentro de 8 segundos volveremos a situarnos en el código del oyente. Durante ese tiempo no hay que hacer ningún cambio en los subtítulos. Crearemos un reloj auxiliar que empiece en la hora 00:00:00,000 y lo pondremos en marcha. Sin embargo, supongamos que un usuario, pausa la reproducción cuando nos encontramos en el tiempo de reproducción 00:00:14,000. Entonces pararemos el *Timer*, pero obtendremos su *initialDelay* que eran 8 segundos. Pararemos el reloj auxiliar y obtenemos la hora que marcaba cuando se pausó el vídeo, que serían las 00:00:02,000, porque han pasado dos segundos desde que se introdujo el subtítulo. Cuando, pasado un tiempo arbitrario, el usuario reanude la reproducción, lo que se debe hacer es crear un nuevo *Timer* cuyo *initialDelay* sea el *initialDelay* del antiguo *Timer* menos la hora que marcaba el reloj auxiliar cuando se pausó la reproducción:  $(8 \text{ segundos} - 2 \text{ segundos}) = 6$  segundos. Este es el tiempo que queda para que vuelva a saltar el oyente y se elimine el subtítulo que se está mostrando actualmente en el área de texto. Esto sucederá entonces en el tiempo de reproducción 00:00:20,000 tal y como estaba previsto.

En la figura 4.27 podemos observar la pantalla principal del reproductor. En ella aparece uno de los fotogramas del vídeo que está reproduciéndose y, bajo éste, el cuadro de texto donde van apareciendo los subtítulos asociados al vídeo. En el panel de control aparecen dos botones, ahora son los de “Stop” y “Pause”, porque la reproducción está en curso. Si la reproducción está parada tan sólo aparecerá el botón de “Play”. Y si la reproducción está pausada aparecerán los botones de “Stop” y “Resume”. En el lado derecho del panel de control aparece el reloj que nos va indicando los tiempos de reproducción.

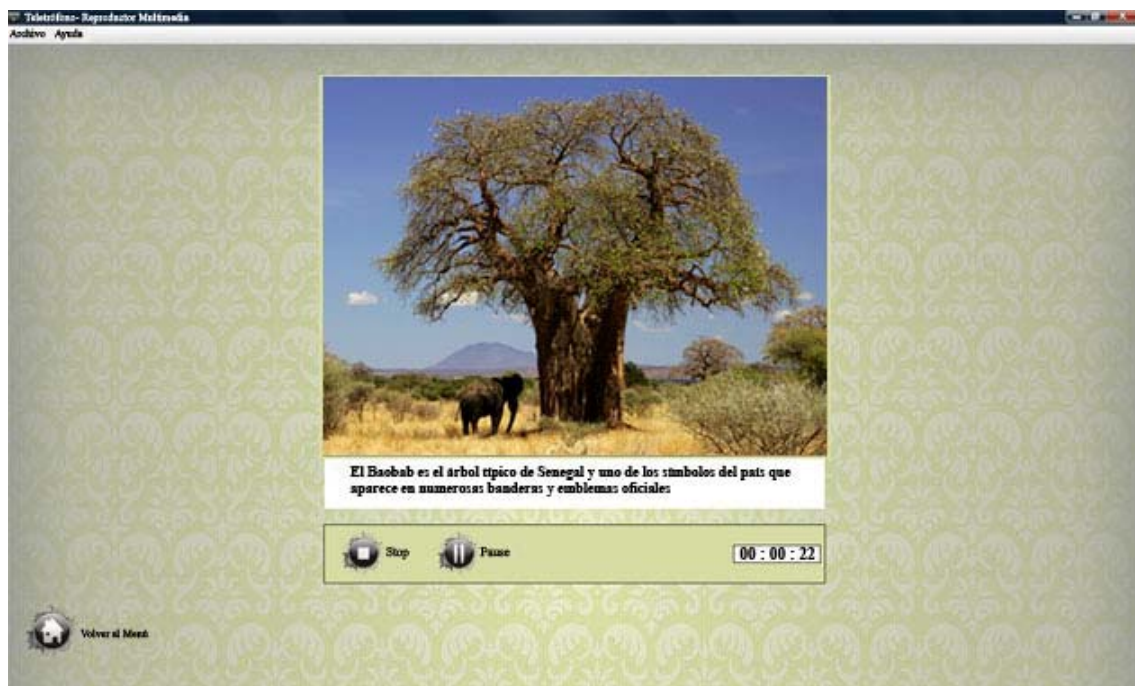


Figura 4.27. Ventana del Reproductor Teletrófono.

Podemos ver que en la parte inferior izquierda de la ventana aparece otro botón. Este botón nos permite cerrar el reproductor y volver a la pantalla del menú principal. Asimismo, podemos ver que se ha añadido a la ventana del reproductor una barra de menú idéntica a la que estaba presente en el menú principal. Dicha barra nos permite salir del programa, acceder al manual de ayuda y ver el cuadro de diálogo ‘Acerca de...’.

#### **4.2.3.2.- El Editor de Subtítulos Teletrófono**

Otra de las herramientas que nos ofrece Teletrófono es el Editor de Subtítulos. Se puede acceder a esta herramienta a través del último icono que aparece en el menú principal en el que figura la frase: “Editar y Re-sincronizar Subtítulos”. Cuando el usuario escoge esta opción desde el menú, la aplicación lanza automáticamente la clase ‘**Editor01**’. Esta clase muestra un diálogo en el que se le pide al usuario la ruta en la que se encuentra el archivo de vídeo cuyos subtítulos se desean editar. El usuario debe pulsar un botón punteado para introducir la ruta a través de un diálogo selector de archivos que, de nuevo, estará implementado por la clase ‘ChooserAbrir’ a la que se le indicará que deseamos un selector que muestre por defecto únicamente archivos de vídeo. Esto se implementaba con la clase ‘FiltroVídeo’ tal y como se explicó para el Reproductor Teletrófono.

Una vez el usuario ha introducido la ruta en la que se encuentra el archivo de vídeo cuyos subtítulos quiere editar, la clase ‘Editor01’ pasará a realizar una serie de comprobaciones. Hay que tener en cuenta que el editor de subtítulos ha sido diseñado especialmente para tratar con vídeos y archivos de subtítulos que hayan sido creados a través de Teletrófono. Por ello, las características que se comprueban en la clase que nos ocupa son las siguientes:

- Se comprueba que el archivo de vídeo introducido realmente exista y se pueda leer.
- Se comprueba que el archivo de vídeo indicado lleve asociada una extensión ‘.avi’. Esto se debe a que, como veremos más adelante, las sesiones de videoconferencia que se graben mediante Teletrófono, serán parcialmente (el vídeo sin audio) almacenadas en este formato. Por tanto, como esta herramienta de edición ha sido creada exclusivamente para tratar grabaciones creadas con nuestro programa, el editor únicamente soportará vídeos con esa extensión.
- Se comprueba que el archivo de vídeo indicado lleve asociado un archivo de audio con la extensión ‘.gsm’. Esto se debe a que, como veremos más adelante, las sesiones de videoconferencia que se graben mediante Teletrófono, serán parcialmente (sólo el audio) almacenadas en este formato. Por tanto, como esta herramienta de edición ha sido creada exclusivamente para tratar grabaciones creadas con nuestro programa, el editor exige que exista este archivo de audio. Al igual que sucedía en el Reproductor Teletrófono, consideraremos que el archivo AVI escogido por el usuario tiene un archivo de audio asociado cuando exista un archivo con la extensión ‘.gsm’ que se pueda leer, ubicado en el mismo directorio que el archivo ‘.avi’ y que se llame igual que éste, a excepción de la extensión del archivo.
- Se comprueba que el archivo de vídeo indicado tenga subtítulos asociados. En este caso, es un requisito indispensable, pues la herramienta sirve para editar y re-

sincronizar los subtítulos del vídeo que escojamos. Para que se considere que un vídeo tiene un archivo de subtítulos asociado, deberá existir un archivo SRT, que se pueda leer, ubicado en el mismo directorio que el archivo de vídeo y que se llame igual que éste, a excepción de la extensión del archivo.

Una vez se ha comprobado que el archivo de vídeo indicado por el usuario cumple todos los anteriores requisitos, ‘Editor01’ lanza la clase ‘**Editor02**’. Esta clase hereda de la clase ‘*JFrame*’ de Java, por lo que será la clase que muestre la pantalla de edición de subtítulos con la que interactuará el usuario. Fijémonos entonces en la pantalla del Editor Teletrófono que se muestra en la figura 4.28.



Figura 4.28. Ventana del Editor Teletrófono.

En la parte superior izquierda de la ventana del editor aparece un cuadro con el texto: “Pulse Play para reproducir el vídeo escogido”. Se trata de un reproductor, idéntico al que hemos implementado en la opción del menú “Reproducir Grabación”.

En la parte central de la pantalla aparece una lista con los subtítulos asociados al vídeo que hemos escogido. En esa lista aparecen, por orden, el texto de cada uno de los subtítulos con sus tiempos de inicio y finalización. Bajo dicha lista se sitúa un botón que se llama “Editar”.

Vemos que la ventana posee también una barra de menú idéntica a la que estaba presente tanto en el menú principal, como en el Reproductor. Dicha barra nos permite salir del programa, acceder al manual de ayuda y ver el cuadro de diálogo ‘Acerca de...’.

Por último aparece un botón en la esquina inferior izquierda que permite al usuario cerrar la ventana del Editor de Subtítulos y volver al menú principal.

Con esta interfaz se pretende que el usuario, en cuanto acceda al editor, lo primero que pueda ver sean los subtítulos asociados al vídeo escogido con los tiempos de inicio y fin de cada uno. Cuando desee comprobar cómo quedan realmente esos subtítulos en el vídeo, tan sólo tendrá que pulsar el botón de “Play” del reproductor. Entonces comenzará la reproducción del vídeo y de los subtítulos. A su vez, el usuario podrá

editar el subtítulo que desee, seleccionándolo en la lista y pulsando a continuación el botón “Editar”. Esto provocará la aparición en pantalla de un nuevo panel, el panel de edición. En el panel de edición aparecerán diversos campos de texto con las horas, minutos y segundos de inicio y fin del subtítulo así como un campo de texto, de mayor longitud, con el propio texto del subtítulo. Todos esos campos podrán editarse, de modo que el usuario se puede posicionar en ellos y cambiarlos a su antojo. Una vez haya realizado los cambios deseados, el usuario deberá pulsar un botón, también situado en este panel que se llama “Confirmar cambios en el subtítulo”. El panel de edición desaparecerá de nuevo y el archivo que contiene los subtítulos se habrá modificado. Se actualizará la lista de subtítulos que aparece en el Editor y podremos ver como han quedado los cambios realizados a través del reproductor que contiene el propio Editor.



Figura 4.29. Ventana del Editor Teletéfono, editando un subtítulo.

En la figura 4.29 se puede ver cómo queda la pantalla del Editor de Subtítulos mientras el usuario visualiza el vídeo con los subtítulos y modifica el segundo de ellos. Sabemos que el subtítulo que está siendo modificado es el segundo, porque es el que aparece subrayado en azul en la lista.

Una vez hemos visualizado el Editor Teletéfono, pasemos a describir cómo de han implementado cada uno de sus componentes.

Habíamos comentado, que la clase que ejecuta esta pantalla es la clase ‘Editor02’. Pues bien, dicha clase, antes de crear todos los componentes que aparecen en la pantalla y que conforman la interfaz gráfica, realiza una comprobación. Se trata de una comprobación adicional a las que realizó la clase ‘Editor01’. El Editor necesita asegurarse de que el vídeo que se le ha pasado como argumento a su constructor tenga como dimensiones máximas 352x288 píxeles. Esto sucede, de nuevo, porque como hemos dicho el editor está diseñado para trabajar con vídeos y subtítulos que hayan sido creados a través de la herramienta de grabación de videoconferencias de Teletéfono y, como se explicará más adelante, dichos vídeos tendrán una dimensiones máximas de 352x288 píxeles. Pero, ¿cómo realiza el Editor dicha comprobación? La clase ‘Editor02’ creará siempre por defecto un *Player* de vídeo:



```
Manager.createRealizedPlayer(new MediaLocator(new File(rutaArchivo).toURL()));
```

Del mismo modo que procedimos en el Reproductor Teletrófono, creamos un *MediaLocator* que haga referencia a la ubicación del archivo de vídeo. Después pasamos ese localizador al método del *Manager createRealizedPlayer*. Después deberemos obtener el componente visual del *Player* ejecutando la sentencia:

```
Component comp = player.getVisualComponent();
```

Ahora, antes de añadir dicho componente visual al panel de nuestro editor, comprobaremos las dimensiones del vídeo, haciendo:

```
int ancho= comp.getPreferredSize().width;
```

```
int alto = comp.getPreferredSize().height;
```

Comprobaremos entonces si se cumple la condición:

```
if((ancho> 352) || (alto > 288)){ (...)
```

Si se cumple esta condición, el vídeo no se reproducirá. Se mostrará un mensaje por pantalla al usuario indicándole que las medidas del vídeo no pueden superar los 352x288 píxeles. Se cerrará el editor y se volverá al menú principal de Teletrófono.

Si por el contrario las dimensiones del vídeo son menores o iguales a 352x288, se añadirá el componente visual del *Player* al panel correspondiente del editor, se creará un *Player* para el archivo de audio ‘.gsm’ (tal y como se explicó para el Reproductor Teletrófono) y se mostrará la ventana de edición que hemos visto en la figura 4.28.

El primer elemento que aparece en la pantalla de edición es un reproductor. Dicho reproductor se ha implementado de forma idéntica a la que se implementó el Reproductor Teletrófono al que se accedía desde el menú principal de la aplicación. Ambos reproductores muestran al usuario el vídeo escogido con sus subtítulos. En vez de usar los controles propios de un *Player* JMF, en estos reproductores se han implementado controles propios para comenzar, pausar, parar y reanudar la reproducción y se ha implementado un reloj propio para visualizar el tiempo de reproducción. La única diferencia entre ambos, es que el Reproductor Teletrófono al que se accede directamente desde el menú principal, admite varios formatos y medidas de vídeo e incluso reproduce vídeos sin subtítulos; mientras que el reproductor embebido dentro del Editor Teletrófono tan sólo admite vídeos en formato AVI con unas dimensiones menores o iguales a 352x288 píxeles, que lleven un archivo de audio ‘.gsm’ y un archivo de subtítulos ‘.srt’ asociados.

El segundo elemento que aparece en la pantalla de edición es la lista con los subtítulos asociados al vídeo escogido. El reproductor que aparece en esta ventana, nada más comenzar la clase ‘Editor02’, inicia la lectura del documento de texto con los subtítulos para importar dichos subtítulos y sus tiempos de inicio y fin a variables locales tipo *array*. Esto lo hacemos mediante un método que hemos llamado *importarDatos*. Para crear la lista en la ventana con los subtítulos, tan sólo hizo falta entonces crear un objeto de tipo *List* e ir añadiendo a esta lista cadenas de texto formadas con los datos que contenían los *arrays* con los datos importados. Un ejemplo de un elemento de esta lista que puede ver el usuario nada más iniciar el editor sería:

```
00:00:01,000 --> 00:00:10,000 --> ¿Adivinais en qué país estamos?
```

, el primer tiempo que aparece en cada línea de la lista es el tiempo de inicio del subtítulo, después aparece el tiempo de finalización y por último el texto en sí. Estos son los tres elementos que el usuario va a poder modificar a través del editor.



El usuario podrá modificar un subtítulo seleccionándolo en la lista (la línea de la lista en la que se encuentra el subtítulo seleccionado se subrayará automáticamente en color azul) y pulsando a continuación el botón “Editar”. Esta acción hará que aparezca un panel de edición, que antes no era visible, en la parte inferior de la ventana. En el panel de edición aparecen diversos campos de texto (*TextField*) que el usuario podrá modificar, con los tiempos de inicio y fin del subtítulo escogido en la lista y el texto de dicho subtítulo. Cuando el usuario haya realizado todos los cambios en el subtítulo deberá pulsar el botón “Confirmar cambios en el subtítulo”. Si pulsamos dicho botón mientras estamos reproduciendo el vídeo, aparecerá por pantalla un mensaje al usuario informándole de que para ejecutar los cambios es necesario que pare la reproducción del vídeo. El usuario pulsaría entonces el botón “Stop” del reproductor y ya podría pulsar el botón de confirmación de los cambios. Esta condición se exige para evitar problemas de actualización de subtítulos en el reproductor.

En cada uno de los *TextField* que componen el panel de edición se puede introducir un número limitado de caracteres. Por ejemplo, el campo de texto que contiene el minuto de inicio del subtítulo no puede tener más de dos caracteres o el campo con el texto del subtítulo no puede sobrepasar los ciento veinticinco caracteres. Para asegurarnos de que el usuario no introduzca más caracteres de los debidos se creó una clase que nos será de gran utilidad a lo largo del desarrollo de todo el programa, es la clase ‘**TamañoDocumento**’. Esta clase hereda de la clase de java ‘*PlainDocument*’ y tiene como único atributo un número entero que indicaría el número máximo de caracteres que queremos para el documento. Tan sólo se ha definido un método que sobre-escribe el método *insertString* de la clase ‘*PlainDocument*’ y que obliga a la cadena de texto que se le pasa como parámetro a no superar el límite establecido. En caso de que lo supere, truncamos la cadena. A través de esta clase, logramos que los campos de texto del panel de edición en los que el usuario puede escribir no permitan más de un número determinado de caracteres. Por ejemplo, para que en el campo donde el usuario puede cambiar el propio texto del subtítulo no se puedan insertar más de ciento veinticinco caracteres, hacemos:

```
subtitulo.setDocument(new TamañoDocumento(125));
```

, siendo ‘subtitulo’ el nombre del citado *TextField*.

Nada más pulsarse el botón que confirma los cambios en el subtítulo seleccionado, la clase ‘Editor02’ pasa a hacer varias comprobaciones:

- Sabemos que el número de caracteres que el usuario ha escrito en cada uno de los campos de texto no es superior al número de caracteres permitidos. Sin embargo, el usuario puede haber introducido menos caracteres de los deseados, lo cual puede dar lugar a errores. Lo primero que se hace es, por tanto, comprobar que el número de caracteres introducidos por el usuario en cada *TextField* es justo el número deseado.
- Se comprueba que ninguno de los caracteres introducidos en los campos de texto que conforman las horas de inicio y fin del subtítulo, sea un espacio en blanco.
- Se comprueba que todos los caracteres introducidos en los campos de texto que conforman las horas de inicio y fin de subtítulo, sean un número. Esta comprobación la implementamos mediante un sencillo método al que recurrimos en varias ocasiones para implementar el programa. El método recibe como parámetro una cadena de texto y devuelve un *booleano* que será verdadero si la

cadena de texto pasada en un número. Devolverá falso en el caso contrario. Para hacer esta comprobación basta con tratar de ejecutar la sentencia:

```
Integer.parseInt(cadena);
```

, si esta expresión hace saltar una excepción del tipo *NumberFormatException*, significará que la cadena de texto pasada como parámetro no representa un número.

- Además comprobaremos aquellos números introducidos por el usuario que indiquen una hora, no sean mayores de veintitrés, y que aquellos que indiquen minutos y segundos no sean mayores de cincuenta y nueve.

Por último se realizarán una serie de comprobaciones relativas a los tiempos de inicio y fin del subtítulo que el usuario ha introducido. Para estas comprobaciones nos serviremos continuamente de objetos de tipo *Time* (esta vez del paquete *sql* de Java), que creamos a partir de las cadenas de texto introducidas por el usuario. Concretamente usaremos el constructor:

```
Time t = new Time(int hora, int minutos, int segundos);
```

Usaremos los métodos de comparación *after* y *equals*, que son realmente sencillos de utilizar. Estos métodos los hereda la clase '*Time*' de la clase '*Date*' (del paquete *util* de Java). Las comprobaciones de esta índole que se realizan son las siguientes:

- Se comprueba que el tiempo de inicio del subtítulo sea anterior al tiempo de finalización del mismo.
- El tiempo de finalización del subtítulo deberá sea anterior al tiempo en que el vídeo finaliza. Es decir, todos los subtítulos deben finalizar antes de que el vídeo termine.
- Comprobaremos que el tiempo de fin del subtítulo sea anterior al tiempo de inicio del siguiente subtítulo.
- Por último comprobaremos que el tiempo de inicio del subtítulo sea mayor que el tiempo de fin del subtítulo anterior.

Una vez se han llevado a cabo toda esta serie de comprobaciones hay que realizar los cambios en el fichero de texto con los subtítulos. Para ello, deberemos servirnos de nuevo de los objetos *BufferedReader* y *FileReader* y seguir el mismo proceso de lectura de archivo que explicamos cuando importamos los subtítulos para el Reproductor Teletrófono. Sin embargo, esta vez nuestro objetivo es diferente. Ahora no pretendemos leer el fichero con los subtítulos para extraer datos, sino que primero debemos buscar en el archivo las líneas de texto correspondientes al subtítulo que queremos modificar; una vez encontradas deberemos re-escribir la línea de tiempos y de subtítulos con los nuevos datos que el usuario ha introducido en los campos de texto del panel de edición.

Por tanto, en primer lugar lo que hacemos es leer el fichero de texto con los subtítulos buscando la línea del documento en la que aparecen los tiempos de inicio y fin del subtítulo a modificar y la línea del documento que contiene el inicio del propio texto del subtítulo a modificar. Es importante reseñar, que lo que realmente se busca cuando recorremos el documento, es aquella línea que introduce/presenta el subtítulo buscado, que será una línea que únicamente contiene el número de subtítulo. Por ejemplo, si el usuario ha modificado el subtítulo que a él le aparece como segundo en la lista, lo que la aplicación busca en el documento son los tiempos y texto que aparezcan tras una línea que sólo contiene el número '2'. Veámoslo gráficamente en la figura 4.30.

Como lo que buscamos es el número de subtítulo, es muy importante que la numeración de los subtítulos en el archivo sea correcta. Si recordamos, cuando explicábamos la implementación del Reproductor Teletrófono comentábamos que no importaba que en el documento de texto la numeración de los subtítulos fuese incorrecta (siempre y cuando el minutaje fuese correcto), porque la reproducción de éstos se realizaría por orden de aparición. En el Editor Teletrófono no ocurre lo mismo. Si la numeración en el archivo de subtítulos fuese errónea, el editor tendría problemas a la hora de modificar los subtítulos. Por ello se recomienda, una vez más, que el archivo que contiene los subtítulos sea manipulado únicamente a través de Teletrófono. Evitaremos así errores indeseados. Tras este paréntesis, continuamos explicando cómo se modifican los subtítulos.

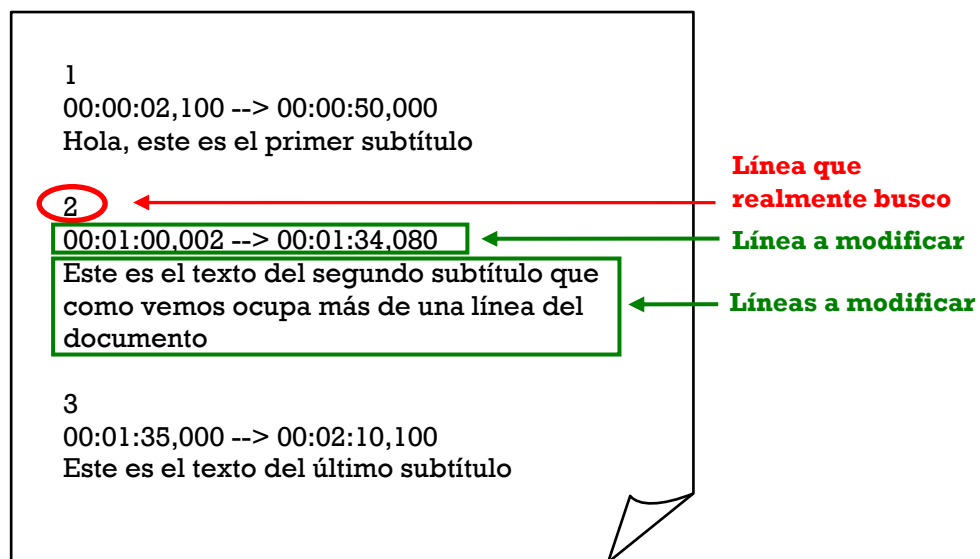


Figura 4.30. Funcionamiento de la edición de subtítulos.

Necesitamos modificar el fichero con los subtítulos para que así refleje los cambios realizados por el usuario. Pero lo que vamos a hacer en realidad no es modificar el fichero existente, sino escribir uno nuevo que combine el antiguo con los nuevos datos introducidos por el usuario. No es posible modificar mediante Java una línea o unos caracteres concretos del texto, sino que es necesario rescribir el archivo.

Por tanto, una vez encontramos en el documento de texto las líneas a modificar, guardamos en variables locales ambos números de línea y se los pasamos como argumento a un método que hemos llamado *modificarFichero*. Este método va leyendo el fichero con los subtítulos y va escribiendo lo mismo que lee, en un nuevo fichero de texto. Cuando alcanza las líneas a modificar, en vez de copiar el contenido de esas líneas, escribe en el nuevo fichero los nuevos datos que introdujo el usuario en el Editor. Finalmente borra el antiguo fichero y renombra al nuevo fichero. De este modo ya tendremos un nuevo documento de subtítulos con los cambios que el usuario deseaba realizar.

Luego ejecutaremos de nuevo el método *importarDatos*, para actualizar la lista con los subtítulos que aparece en la ventana del editor. Recordamos que este método lo que hacía era leer directamente el fichero de texto con los subtítulos e ir extrayendo los tiempos y el contenido de cada subtítulo para añadirlo a la lista en el formato deseado. Por último ejecutamos el método *actualizarSubtítulos* que se encarga de actualizar, en

base al nuevo fichero de subtítulos, las variables de tipo *array* en las que guardamos tanto los tiempos de inicio y fin de los subtítulos como su texto. De este modo, cuando el usuario reproduzca de nuevo, a través del editor, el vídeo, los subtítulos aparecerán ya modificados.

#### 4.2.3.3.- La Videoconferencia con Teletrófono

Pasemos ahora a explicar cómo se ha implementado la funcionalidad principal de Teletrófono. Recordamos que el objetivo principal de nuestra aplicación era establecer una videoconferencia entre dos extremos alejados entre sí, en un contexto educativo.

Recordamos también que planteábamos una situación bastante definida en la que cada lado de la comunicación jugaba un papel u otro dependiendo de sus características. En la comunicación, un extremo sería denominado extremo local, y el otro extremo sería el remoto. El lado local se caracterizaba por ser un lugar fijo, probablemente un aula de un centro educativo, en el que se encontraba un profesor y un grupo de alumnos. El lado remoto podía ser cualquier lugar, tanto exterior como interior, en el que se encontraba un sólo profesor con la intención de hablar al otro extremo acerca del entorno al que se había desplazado. Explicamos también en apartados anteriores que ambos extremos de la videoconferencia requerían funcionalidades distintas. Es por ello, que quedamos en que lo primero que había que hacer era tener claro qué rol debía jugar cada extremo, para poder luego ejecutar un icono u otro de nuestro menú principal Teletrófono.

Si recordamos la figura 4.23, en la que aparecía la ventana del menú principal de nuestro programa, el primer icono, llamado “Iniciar Videoconferencia desde el extremo Local”, es sobre el que deberá pulsar el profesor que se encuentra en el aula con los alumnos. El profesor situado en un lugar arbitrario, fuera del centro de estudios, deberá por el contrario pulsar sobre el segundo icono del menú, llamado “Iniciar Videoconferencia desde el extremo Remoto”.

En cuanto el usuario pulse cualquiera de las citadas opciones del menú principal le aparecerá una nueva ventana de configuración de la videoconferencia. Estas ventanas serán muy similares para ambos extremos de la comunicación, pero no exactamente iguales. La opción de videoconferencia desde local lanzará la clase ‘**ConfigVideoconfLocal**’, mientras que la opción que escogeremos en remoto lanzará la clase ‘**ConfigVideoconfRemota**’.

Veamos en la figura 4.31 la ventana de configuración de la videoconferencia que nos aparecería si pulsamos sobre la opción “Iniciar Videoconferencia desde el extremo Local”. La ventana aparece dividida en cinco áreas principales: “Título”, “Direcciones”, “Dispositivos de captura”, “Puertos” y un panel con botones.

En el panel “Título” aparece un cuadro de texto (*JTextField*) en el que el usuario puede insertar el nombre que desee dar a la videoconferencia. Este título aparecerá posteriormente en la parte superior de la ventana de videoconferencia. Este campo puede dejarse en blanco si no se desea dar un nombre concreto a la sesión de videoconferencia.

En el panel “Direcciones” tenemos dos campos de texto (*JTextField*). El primer campo de texto muestra la dirección IP de nuestra máquina. Este campo no lo puede modificar el usuario. En el segundo campo de texto el usuario debe insertar la dirección IP del otro extremo de la comunicación. Será necesario que la conozca, si hace falta, poniéndose previamente en contacto con el otro extremo mediante una vía de comunicación adicional.

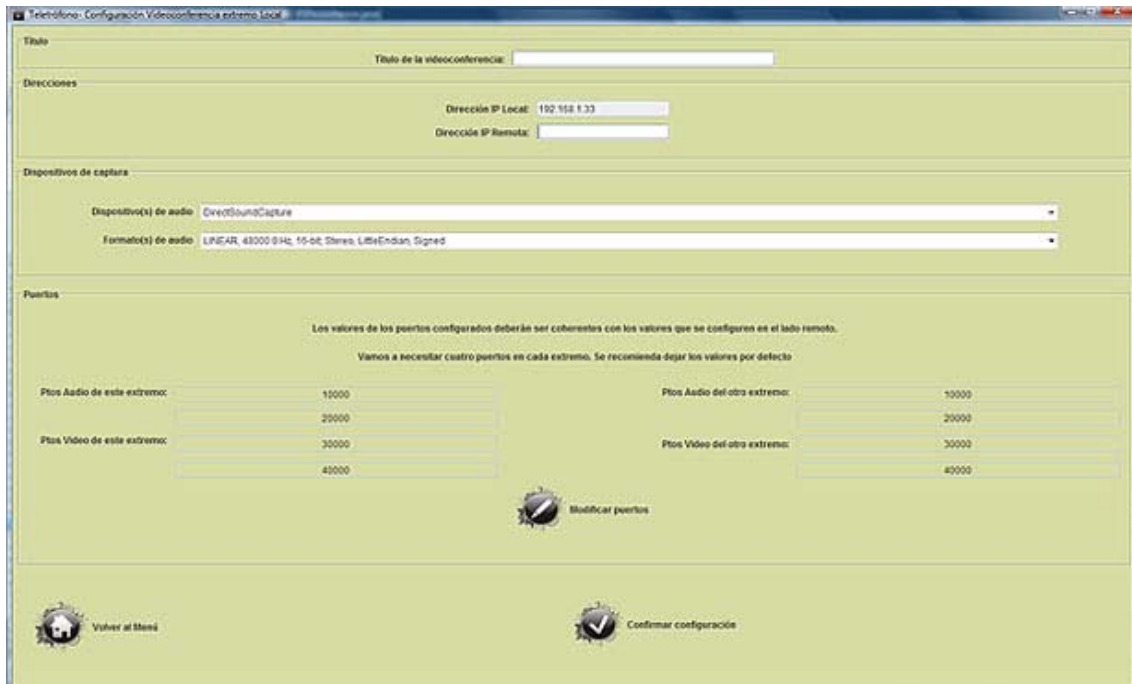


Figura 4.31. Ventana de configuración de la videoconferencia desde el extremo Local.

En el panel “Dispositivos de captura” nos aparecen dos campos de texto que, si pulsamos sobre ellos, despliegan una lista. Este tipo de elemento se implementa mediante la clase ‘*Choice*’ de Java. El primer *choice* nos despliega los dispositivos de captura audio que JMF ha detectado en nuestra máquina. El segundo *choice* nos muestra los formatos de audio en los que se pueden realizar dichas capturas. El usuario podrá escoger, por tanto, el dispositivo con el que quiere que se capture su voz y en qué formato quiere que dicha voz se capture. Siempre se pueden dejar el dispositivo y formato que aparecen por defecto. Recordamos que desde este lado de la comunicación únicamente deseamos capturar audio, pues este será el único tipo de datos que transmitiremos de local a remoto.

Para saber los dispositivos que JMF puede detectar en nuestra máquina deberemos acudir al *CaptureDeviceManager*. Ejecutando la sentencia:

```
Vector devices = CaptureDeviceManager.getDeviceList(null);
```

El método *getDeviceList* devuelve una lista de objetos tipo *CaptureDeviceInfo* correspondientes a los dispositivos detectados en la máquina que pueden capturar datos en el formato que le especifiquemos como parámetro. Si no se especifica ningún formato, como es el caso, el método devuelve una lista con los objetos *CaptureDeviceInfo* de todos los dispositivos de captura disponibles. Obtendremos, por tanto, almacenados en un vector los dispositivos instalados y reconocidos por JMF. Lo

que haremos entonces, será comprobar que este vector no esté vacío, pues ello significaría que no se ha detectado ningún dispositivo de captura. Una vez sabemos que se han detectado dispositivos de captura, debemos escoger aquellos que realmente nos interesan. En este caso buscamos exclusivamente los dispositivos de captura de audio. Para ello ejecutaremos las siguientes sentencias:

```
int deviceCount = devices.size();
Vector audioDevices = new Vector();
for ( int i = 0; i < deviceCount; i++ ) {
    CaptureDeviceInfo cdi = (CaptureDeviceInfo)devices.elementAt(i);
    Format[] formats = cdi.getFormats();
    for ( int j=0; j<formats.length; j++ ) {
        if ( formats[j] instanceof AudioFormat ) {
            audioDevices.addElement(cdi);
            break;
        }
    }
}
```

Lo que hemos hecho a través del código anterior es recorrer el vector que contiene todos los dispositivos de captura instalados y obtener los formatos soportados por cada uno de ellos. Se comprueba si el formato del dispositivo se corresponde con un formato de audio y, en caso de que así sea, se guarda la información de ese dispositivo en un nuevo vector que hemos creado que ya únicamente contiene los dispositivos de captura de audio. Finalmente, para poder mostrar el nombre de esos dispositivos en la lista desplegable de nuestra ventana de configuración de la videoconferencia deberemos hacer:

```
for (int i=0; i<audioDevices.size(); i++) {
    cdi = (CaptureDeviceInfo) audioDevices.elementAt(i);
    audioDeviceChoice.add(cdi.getName());
}
```

Mediante estas líneas recorreremos el vector que contiene los dispositivos de captura de audio disponibles, obtenemos el nombre de cada uno y lo vamos añadiendo a la lista desplegable, que se llama en este caso ‘audioDeviceChoice’.

Ahora nos queda añadir elementos a la lista que muestra los formatos de audio disponibles. Una vez que el usuario ha escogido el dispositivo de captura de audio es todo bastante fácil. Deberemos obtener los formatos que soporta el dispositivo de captura que el usuario ha escogido y los guardaremos en un *array* de tipo *Format*. Luego bastará con que recorramos dicho *array* pasando cada formato a una cadena de texto y añadiendo estas cadenas directamente a la lista desplegable, que se ha llamado ‘audioFormatChoice’. Aprovecharemos también la ocasión para crear un vector con esos formatos de audio. Veamos el código:

```
int i = audioDeviceChoice.getSelectedIndex();
if (i!=-1) {
    CaptureDeviceInfo cdi=(CaptureDeviceInfo)audioDevices.elementAt(i);
    if (cdi!=null) {
        Format[] formats = cdi.getFormats();
        Vector audioFormats = new Vector();
        for (int j=0; j<formats.length; j++) {
            audioFormatChoice.add(formats[j].toString());
            audioFormats.addElement(formats[j]);
        }
    }
}
```

```

    }
}

```

En el caso en que no se encuentren dispositivos de captura de audio el programa mostrará un mensaje por pantalla indicando al usuario que no se puede iniciar una videoconferencia en esas condiciones; se cerrará la ventana de configuración y se mostrará de nuevo el menú principal del programa.

En el siguiente panel, el panel “Puertos”, aparece en primer lugar un aviso importante: “Los valores de los puertos configurados deberían ser coherentes con los valores que se configuren en el lado remoto. Vamos a necesitar cuatro puertos en cada extremo. Se recomienda dejar los valores por defecto.”. Y a continuación aparecen unos campos que, a priori, el usuario no puede modificar, en los que pone:

Ptos. Audio de este extremo: 10000	Ptos. Audio del otro extremo: 10000
20000	20000
Ptos. Vídeo de este extremo: 30000	Ptos. Vídeo del otro extremo: 30000
40000	40000

Si recordamos las figuras 4.19 y 4.20, en las que se nos mostraba el esquema de implementación definitivo de nuestra aplicación de videoconferencia, nos percatamos de que necesitamos enviar datos a través de cuatro puertos en cada uno de los lados de la comunicación. Lo que hemos hecho a la hora de configurar la videoconferencia, es proporcionar al usuario unos puertos por defecto para llevar a cabo la comunicación. Por lo que en el panel “Puertos” de la ventana de configuración de la videoconferencia, se muestra una lista con los cuatro puertos que se usarán por defecto tanto en este lado de la comunicación, como en el otro extremo. Es necesario definir ocho puertos en total, cuatro por cada lado. Es muy importante que ambos lados de la comunicación sepan qué puertos va a usar el otro extremo, pues si en esta pantalla de configuración se ponen unos puertos, y en la pantalla de configuración del otro extremo se indican unos puertos de comunicación distintos, la aplicación no funcionará. Es por ello por lo que se recomienda al usuario siempre, dejar los puertos que aparecen por defecto en la pantalla de configuración de la videoconferencia. Si de todos modos el usuario quisiese cambiar los puertos, acción de la cual debería informar al otro extremo para que allí también se modificasen, éste podría hacerlo a través del botón “Modificar puertos”. Este botón simplemente cambia una propiedad de los campos de texto (*JTextField*) en los que aparecen los números de puertos, para que se conviertan en campos cuyo contenido pueda modificar el usuario.

Por último en la ventana de configuración de videoconferencia desde el extremo local aparecen dos botones. El de la esquina inferior izquierda nos permite salir de la ventana de configuración y volver al menú principal de Teletrófono. El botón que se sitúa centrado en la parte inferior de la ventana es el botón “Confirmar configuración”. Este botón lanzará varios métodos que harán una serie de comprobaciones sobre los datos introducidos por el usuario en la pantalla de configuración. Dichos métodos realizan las siguientes acciones:

- Comprobar que el usuario ha rellenado todos los campos que se le pedían. El único campo que no se comprueba que se haya rellenado es el de “título de la videoconferencia”, porque no es un campo obligatorio.
- Comprobar que la dirección IP introducida como dirección de la máquina con la que se quiere conectar, siga un formato válido para este tipo de direcciones. Para ello habrá que comprobar que la dirección IP introducida esté compuesta por cuatro números separados entre sí a través del carácter punto. Para comprobar esto, nos serviremos de una clase del paquete de Java *util*, llamada ‘*StringTokenizer*’. Dicha clase cuenta en cuantos trozos queda dividido el texto que le pasamos como primer parámetro del constructor, por la cadena de texto que se le pasa como segundo parámetro. Suponiendo que tenemos la dirección IP introducida por el usuario en una variable de tipo *String* llamada ‘dirIP’, haríamos:

```
StringTokenizer st = new StringTokenizer(dirIP, ".");
```

Como queremos que nuestra dirección contenga cuatro números, nuestro método encargado de comprobar la dirección, devolverá *false* si no se cumple esa condición:

```
if((st.countTokens() != 4){ return false;}
```

Este mismo método comprobará que ni el primer ni el último carácter de la dirección introducida sean un punto. También comprobará que los restantes caracteres (que no sean los tres puntos que separan los cuatro números que componen la dirección) sean números. Por último comprobará que ninguno de los cuatro números que componen la dirección tenga más de tres cifras.

- Comprobar que los puertos que ha introducido el usuario, en caso de que haya optado por modificarlos, son números entre 0 y 65.534.

Una vez se ha comprobado que todos los datos de configuración son correctos, la clase ‘*ConfigVideoconfLocal*’ llama a la clase ‘***VideoconfLocal***’ que veremos más adelante.

Veamos ahora en la figura 4.32, la ventana de configuración de la videoconferencia que nos aparecería si pulsamos sobre la opción “Iniciar Videoconferencia desde el extremo Remoto”. Como podemos comprobar, se trata de una ventana muy parecida a la que mostramos para el extremo local. Sin embargo esta ventana la implementa la clase ‘*ConfigVideoconflocal*’. Veamos las diferencias entre las ventanas de configuración de ambos lados.

En la pantalla de configuración de la videoconferencia desde el extremo remoto no nos aparece un campo para insertar el título de la videoconferencia, esto sucede porque no es una opción que nos interese. En el lado local sí es posible que el profesor desee insertar un título, pues este título aparecerá luego en la ventana de videoconferencia que visualizarán todos sus alumnos. Sin embargo, en el lado remoto no tendría sentido que apareciese un título en la ventana de videoconferencia, pues es una ventana que sólo va a ver el profesor en remoto.

El panel de “Direcciones” sí está presente en ambas ventanas de configuración. Se trata de un panel idéntico al que vimos para el otro extremo. Tenemos, en primer lugar, un campo de texto que el usuario no puede modificar, en el que aparece la dirección IP de esta máquina, la que juega el papel de extremo remoto. Bajo este campo de texto aparece otro en el que el usuario deberá introducir la dirección IP de la máquina con la que desea comunicarse. Tal y como dijimos, será necesario que la conozca, si hace falta, poniéndose previamente en contacto con el otro extremo mediante una vía de comunicación adicional.



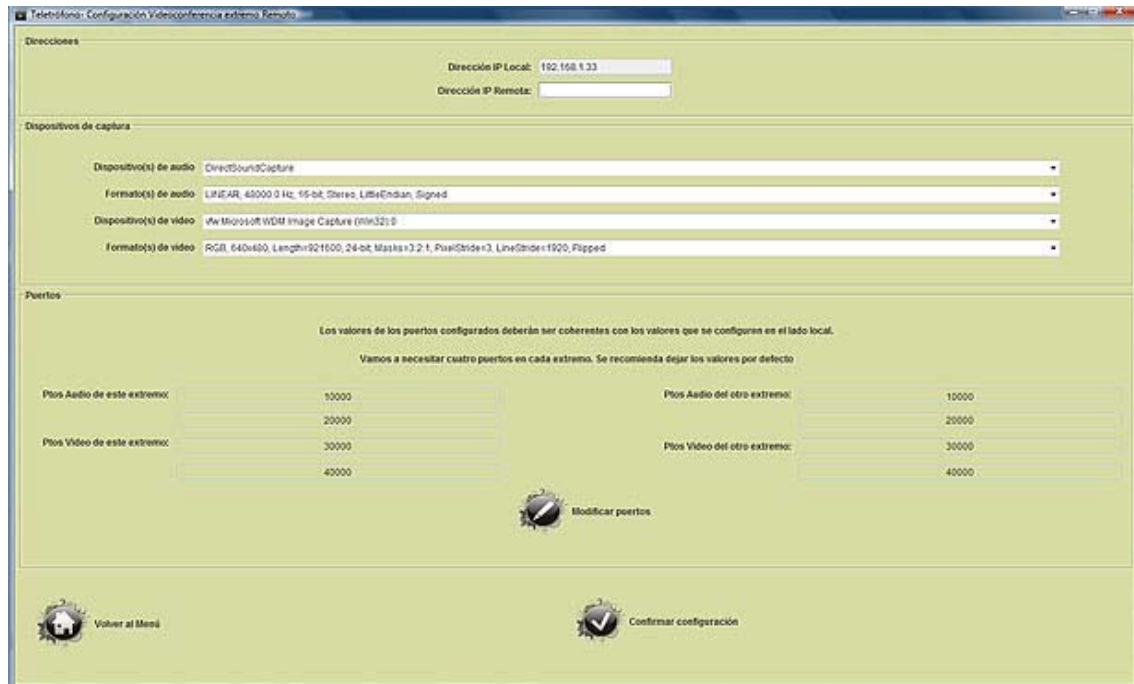


Figura 4.32. Ventana de configuración de la videoconferencia desde el extremo Remoto.

En la ventana de configuración de la videoconferencia para el lado remoto también aparece el panel de “Dispositivos de captura”, pero ahora este panel contiene un par de campos adicionales. Aparte de contener una lista desplegable (*choice*) con los dispositivos de captura de audio disponibles en esta máquina y otra con los formatos en los que se puede capturar la voz del profesor en remoto, aparecen dos nuevas listas desplegables: una con los dispositivos de captura de vídeo disponibles en esta máquina y otra con los formatos en los que se puede capturar dicho vídeo. El profesor en el extremo remoto tendrá, por tanto, total libertad para escoger los dispositivos y formatos de captura que desee. Estos dos campos adicionales aparecen porque recordemos que desde el lado remoto se habrá de transmitir tanto audio como vídeo hacia el lado local de la comunicación, por tanto habrá que usar dos dispositivos de captura, uno para audio y otro para vídeo. Cuando configurábamos la videoconferencia desde el extremo local tan sólo debíamos escoger un dispositivo de captura de audio porque en ese lado no será necesario capturar vídeo.

Para saber y poder mostrar en las listas desplegables los dispositivos de captura de audio y formatos de audio se han seguido exactamente los mismos pasos que se han contado para la configuración de la videoconferencia desde el extremo local. Ahora, para configurarla desde el extremo remoto, hay que seguir los mismos pasos pero además con los dispositivos de captura de vídeo. El código entonces sería:

```
Vector devices = CaptureDeviceManager.getDeviceList(null);
if (devices!=null && devices.size()>0) {
    int deviceCount = devices.size();
    Vector audioDevices = new Vector();
    Vector videoDevices = new Vector();
    for ( int i = 0; i < deviceCount; i++ ) {
        CaptureDeviceInfo cdi = (CaptureDeviceInfo)devices.elementAt(i);
        Format[] formats = cdi.getFormats();
        for ( int j=0; j<formats.length; j++ ) {
            if ( formats[j] instanceof AudioFormat ) {
                audioDevices.addElement(cdi);
            }
        }
    }
}
```

```

        break;
    } else if (formats[j] instanceof VideoFormat ) {
        videoDevices.addElement(cdi);
        break;
    }
}
}
}

```

Ya vimos cómo se rellenaba la lista con los dispositivos de audio. Para añadir a otra de las listas desplegables los dispositivos de vídeo:

```

for (int i=0; i<videoDevices.size(); i++) {
    cdi = (CaptureDeviceInfo) videoDevices.elementAt(i);
    videoDeviceChoice.add(cdi.getName());
}

```

Ya vimos cómo lográbamos mostrar los formatos de audio disponibles, veamos ahora cómo lo hacemos para mostrar los formatos de vídeo en la lista desplegable ‘videoFormatChoice’:

```

int i = videoDeviceChoice.getSelectedIndex();
if (i!=-1) {
    CaptureDeviceInfo cdi=(CaptureDeviceInfo)videoDevices.elementAt(i);
    if (cdi!=null) {
        Format[] formats = cdi.getFormats();
        Vector videoFormats = new Vector();
        for (int j=0; j<formats.length; j++) {
            videoFormatChoice.add(formats[j].toString());
            videoFormats.addElement(formats[j]);
        }
    }
}
}

```

En el caso en que no se encuentren dispositivos de captura de audio y vídeo el programa mostrará un mensaje por pantalla indicando al usuario que no se puede iniciar una videoconferencia en esas condiciones; se cerrará la ventana de configuración y se mostrará de nuevo el menú principal del programa.

Una vez explicada la implementación del panel de “Dispositivos de captura” pasamos a ver el siguiente panel. El panel de “Puertos” de la ventana de configuración de la videoconferencia desde el extremo remoto es idéntico al que aparecía cuando configurábamos la videoconferencia desde el lado local. Aparecerán ocho puertos por defecto, cuatro para cada lado de la comunicación. Y aparecerá también el mensaje de aviso en el que se recomienda que se dejen estos valores. Sin embargo, tal y como ocurría en el lado local, se pone a disposición del usuario un botón que le permita modificar estos valores por defecto si lo desea.

Por último, como vemos en la Figura 4.32, en la parte inferior de la ventana de configuración disponemos de dos botones. Con el que se sitúa a la izquierda podremos cancelar la configuración de la videoconferencia y volver a la pantalla del menú principal de Teletrófono. Con el que está centrado en pantalla el usuario confirmará los datos de configuración que ha introducido. Este botón realizará las mismas comprobaciones que se hacían cuando se pulsaba este mismo botón en la configuración de la videoconferencia desde el lado local; después llamará a la clase ‘**VideoconfRemoto**’.

Una vez revisada la implementación de la configuración de la videoconferencia desde ambos lados de la comunicación, vamos a ver cómo se implementa en la herramienta de videoconferencia, la propia transferencia de datos multimedia.

## •Extremo Local

En primer lugar vamos a echar un vistazo a la interfaz gráfica con la que el usuario debe interactuar a partir de este momento. Comencemos hablando del lado local, una vez el usuario haya introducido los datos correctamente en la ventana de configuración y haya pulsado el botón de confirmación, le aparecerá una ventana como ésta:



Figura 4.33. Ventana de inicio de videoconferencia del extremo Local.

Esta ventana la implementa la clase 'VideoconfLocal', que será la clase principal que llevará a cabo la videoconferencia. Tal y como veremos más adelante, desde la clase 'VideoconfLocal' iremos llamando, cuando corresponda, a otras clases de la aplicación que implementarán, cada una de ellas, una porción de la comunicación. Todas juntas, harán posible la videoconferencia.

En primer término de la ventana de videoconferencia podemos ver un letrero que indica al usuario el botón que debe pulsar cuando desee iniciar la videoconferencia. Este letrero tan sólo aparece mientras que el usuario no pulse el botón que inicia la videoconferencia. Cuando la comunicación entre los extremos se haya establecido, en vez de este letrero, nos aparecerá la imagen del interlocutor en remoto.

Bajo este aviso aparece un campo de texto de color verde, vacío. Este será el campo de texto donde vayan apareciendo los subtítulos que desde esta misma pantalla el usuario irá introduciendo, como veremos después. En el campo de texto que acabamos de mencionar el usuario no puede escribir directamente.

En la parte inferior de la pantalla aparecen una serie de botones. El primero de ellos, que ha aparecido en otras ocasiones, permite al usuario volver a la pantalla del menú principal de Teletrófono. A continuación tenemos un botón con un icono de un teléfono que se llama “Iniciar Videoconferencia”. Este botón será el que el usuario debe pulsar cuando quiera comenzar la comunicación con el otro extremo. A la derecha de éste aparece un botón, deshabilitado por el momento, que se llama “Finalizar Videoconferencia”. Este botón se habilitará cuando se haya establecido la comunicación con el lado remoto. Por último aparecen otros dos botones: “Mostrar panel de inserción de subtítulos” y “Ocultar panel de inserción de subtítulos”. Si pulsamos el primero de ellos, aparecerá bajo éste un nuevo panel que contendrá: un campo de texto, en el que el usuario podrá insertar el rótulo que desee, y un botón, que hará que el texto introducido se posicione en el campo de texto situado bajo la imagen que recibimos del otro extremo (ahora ocupada con un letrero de aviso mientras no iniciemos la videoconferencia). El otro botón mencionado simplemente oculta este panel para evitar que haya demasiados elementos en pantalla cuando no estamos insertando nuevos subtítulos.

Cuando el usuario haya pulsado el botón “Iniciar Videoconferencia” y se haya establecido la comunicación, la ventana de Teletrófono tendrá un aspecto como el que muestra la figura 4.34.



Figura 4.34.Videoconferencia Teletrófono desde el lado Local.

El usuario en el lado local de la comunicación, mediante la videoconferencia Teletrófono, será capaz de ver al profesor situado en el extremo remoto y de mantener una comunicación bidireccional de audio con él. Además podrá introducir subtítulos y grabar la sesión.

Cuando se establece la comunicación vemos, en la anterior figura, que aparece un nuevo botón, llamado “Comenzar grabación”, al final del panel de botones. En cuanto el usuario pulse este botón, se comenzará a almacenar en nuestra máquina la imagen y el audio que estamos recibiendo desde el otro extremo. También se almacenará, en un archivo SRT, cada uno de los subtítulos que a partir de este momento el usuario en el



lado local vaya introduciendo. Mientras se está grabando la videoconferencia, el botón para comenzar la grabación quedará sustituido por el botón “Parar grabación”. Si el usuario pulsa este último botón se detendrá la grabación de la videoconferencia y de los subtítulos. Sólo se permite realizar una grabación por sesión de videoconferencia. Por tanto, una vez que se ha pulsado el botón que finaliza la grabación no se podrá volver a grabar ningún flujo de datos de la sesión.

Como vamos a comprobar, el modo de introducir subtítulos es muy sencillo. El usuario deberá pulsar el botón “Mostrar panel de inserción de subtítulos”, introducir el subtítulo deseado en el nuevo campo de texto que le aparece y pulsar el botón de aceptar. Automáticamente ese texto que ha introducido aparecerá en el campo de texto que se sitúa bajo la imagen que recibimos desde remoto y desaparecerá del campo de texto en el que lo hemos introducido. En la figura 4.35 podemos ver un detalle de la ventana de videoconferencia del lado local. En ella se puede ver el panel de inserción en el que el usuario acaba de escribir un texto que quiere introducir como subtítulo:



Figura 4.35. Inserción de subtítulo (I).

A continuación el usuario deberá pulsar el botón ‘+’ y, si lo desea, ocultar de nuevo el panel de inserción. Quedando entonces la ventana de videoconferencia como se muestra en la figura 4.36.



Figura 4.36. Inserción de subtítulo (II).

Veamos cómo hacemos para que la herramienta de videoconferencia funcione correctamente y cómo se ha implementado toda la funcionalidad que ésta lleva asociada.

En cuanto el usuario pulsa el botón “Iniciar Videoconferencia”, la clase ‘VideoconfLocal’ crea una instancia de la clase ‘**Espere**’. La clase ‘Espere’ hereda de la clase ‘JFrame’ y simplemente muestra una ventana sin marco que contiene un icono de un reloj de Arena y la frase: “Espere... Iniciando la videoconferencia”. La aplicación mostrará esta ventana hasta que la comunicación se haya establecido por completo entre el extremo local y el remoto.

Lo siguiente que hará el oyente del botón “Iniciar Videoconferencia” será almacenar en variables locales los valores de configuración de la videoconferencia que el usuario fijó, para este extremo, en la ventana de configuración. Es importante tener ahora en cuenta el esquema de implementación que se plasmó en las figuras 4.19 y 4.20. En esas figuras se ve cómo para llevar a cabo una videoconferencia que cumpla los objetivos y requisitos definidos íbamos a hacer uso finalmente de cuatro puertos en cada uno de los extremos de la comunicación. Por ello, en la ventana de configuración de la videoconferencia, sea en el lado remoto o en el local, aparecen los valores de los ocho puertos que se tendrán que utilizar en total. Volviendo a las acciones que lleva a cabo el oyente del botón que inicia la videoconferencia, tenemos que decir que, en concreto, crearemos cuatro variables de tipo *array* preparadas para contener cadenas de texto (*String*). Cada una de estas variables contendrá tres cadenas de texto, como sigue:

- La variable ‘destinoAudio’ contendrá el primer puerto local para el audio, la dirección IP remota y el primer puerto remoto para el audio. Usando solamente los datos que contiene esta variable deberemos ser capaces de establecer una comunicación bidireccional de audio y reproducir en este lado el audio recibido. Para realizar esta tarea nos serviremos principalmente de la clase ‘**AudioLocal**’.
- La variable ‘destinoVideo’ contendrá el primer puerto local para el vídeo, la dirección IP remota y el primer puerto remoto para el vídeo. Usando solamente los datos que contiene esta variable deberemos ser capaces de recibir vídeo desde el lado remoto y reproducirlo en local. Para realizar esta tarea nos serviremos principalmente de la clase ‘**ReceptorVideo**’.
- La variable ‘destinoAudio2’ contendrá el segundo puerto local para el audio, la dirección IP remota y el segundo puerto remoto para el audio. Usando solamente los datos que contiene esta variable deberemos ser capaces de recibir audio del extremo remoto y almacenar este audio en un archivo. Para realizar esta tarea nos serviremos principalmente de la clase ‘**AudioLocal2**’.
- La variable ‘destinoVideo2’ contendrá el segundo puerto local para el vídeo, la dirección IP remota y el segundo puerto remoto para el vídeo. Usando solamente los datos que contiene esta variable deberemos ser capaces de recibir vídeo del extremo remoto y almacenar este vídeo en un archivo. Para realizar esta tarea nos serviremos principalmente de la clase ‘**ReceptorVideo2**’.

Como vemos las dos primeras variables las usaremos para llevar a cabo la propia comunicación de audio y vídeo que componen la videoconferencia, entre los dos extremos; mientras que las últimas dos variables las necesitaremos para recibir un flujo adicional de audio, y otro de vídeo, que podamos ir almacenando en nuestra máquina en caso de que el usuario desee grabar la sesión.

En la clase ‘VideoconfLocal’ también se crearán variables locales que contengan el dispositivo de captura de audio que el usuario escogió así como el formato de audio que eligió. Una vez hemos creado todas estas variables con los datos de configuración introducidos por el usuario, el oyente del botón que inicia la videoconferencia, crea una nueva instancia de la clase ‘ReceptorVideo’.

La clase ‘ReceptorVideo’ tiene como misión recibir el flujo de vídeo que se envíe desde el extremo remoto a través de una sesión RTP, para posteriormente reproducirlo en este lado de la comunicación junto con el flujo de audio que recibiremos a través de ‘AudioLocal’. Como la clase establecerá una conexión RTP, necesitaremos recibir notificación de los eventos de este protocolo, por este motivo, esta clase implementa las clases abstractas de Java ‘*ReceiveStreamListener*’, ‘*SessionListener*’ y ‘*RemoteListener*’. Su constructor recibe como parámetros, el *array* ‘destinoVideo’, el *array* ‘destinoAudio’ y el dispositivo y formato de captura de audio que el usuario escogió. La principal función entonces de esta clase es establecer una sesión RTP a través de la cual recibir vídeo del otro extremo. Recordemos que el principal objeto que ha de usarse para coordinar una sesión RTP es el *RTPManager*, habrá que crear, por tanto, un objeto de este tipo y asociarle los oyentes que implementa nuestra clase:

```
RTPManager mgr = RTPManager.newInstance();
mgr.addSessionListener(this);
mgr.addReceiveStreamListener(this);
mgr.addRemoteListener(this);
```

Ahora debemos crear un par de objetos del tipo *SessionAddress* que quedarán definidos cuando les indiquemos una dirección IP y un puerto a cada uno. Crearemos un *SessionAddress* con la dirección IP local y el primer puerto local para vídeo; y crearemos otro objeto de este tipo con la dirección IP del otro extremo y el primer puerto remoto para vídeo. Recordamos que la variable ‘destinoVideo’ contiene: [Puerto Local Vídeo 1, dirección IP Remota, Puerto Remoto Vídeo 1].

```
int PtoVideoLocal1= new Integer(destinoVideo[0]).intValue();
int PtoVideoRemoto1= new Integer(destinoVideo[2]).intValue();
InetAddress ipAddrRemota = InetAddress.getByName(destinoVideo[1]);
InetAddress ipAddrLocal = InetAddress.getLocalHost();
SessionAddress localAddr = new SessionAddress(ipAddrLocal, PtoVideoLocal1);
SessionAddress destAddr=new SessionAddress( ipAddrRemota, PtoVideoRemoto1);
```

Ahora ya tenemos todos los objetos necesarios creados para iniciar la sesión RTP. Llamaremos al método del administrador que inicializa la sesión. Después al método que abre la sesión, al que le pasamos la dirección y puerto del lado remoto, que comenzará a generar informes de control RTCP.

```
mgr.initialize(localAddr);
mgr.addTarget(destAddr);
```

Deberemos implementar los métodos *update* del oyente *SessionListener*, *ReceiveStreamListener* y *RemoteListener*. El método *update* del oyente *SessionListener* lleva asociado un evento *SessionEvent* al que llamaremos ‘event’, así que lo que haremos será comprobar si ese evento es un caso de un *NewParticipantEvent*. En ese caso ejecutaremos la sentencia:

```
Participant p = ((NewParticipantEvent)event).getParticipant() ;
```

Así obtenemos una instancia de una clase que implementa la interfaz *Participant*, que en este caso representa al participante en la sesión RTP del otro extremo.

Ahora vamos a ver lo que hará el método *update* del oyente *RemoteListener*. Este método lleva asociado un evento *RemoteEvent* al que llamaremos “event”, así que realizaremos una serie de acciones en el caso en que este evento sea un caso de un *ReceiverReportEvent* o de un *SenderReportEvent*. Simplemente almacenaremos temporalmente algunos de los datos de control (como nombre canónico del participante del otro extremo, el identificador de fuente de sincronización, la hora exacta del evento) que nos proporciona el informe asociado a este evento, que obtenemos con la sentencia:

```
ReceiverReport rr= ((ReceiverReportEvent) event).getReport();
```

Por último, veamos las acciones que debe implementar el método *update* del oyente *ReceiveStreamListener*. Este método es el que lleva a cabo las acciones clave que debe realizar la clase ‘ReceptorVideo’. Este método lleva asociado un evento *ReceiveStreamEvent* al que llamaremos ‘event’. Comprobaremos qué tipo de *ReceiveStreamEvent* es. Antes extraeremos una serie de datos del evento, como el gestor dueño del evento y el participante que generó el evento:

```
RTPManager mgr = (RTPManager)event.getSource();  
Participant participant = event.getParticipant();
```

Vamos a explicar únicamente como se ha actuado en los tipos de evento que realmente nos interesan. Por ejemplo, si el evento es del tipo *ByeEvent*, además de extraer de éste detalles, como la razón por la que el participante del otro extremo abandona la sesión, el nombre de éste o el identificador de fuente de sincronización, se mostrará al usuario una ventana, implementada con la clase ‘DiálogoEstandar’ que ya mencionamos en otras ocasiones, en la que se le informará de que el usuario con el que estaba conectado ha abandonado la sesión. La ventana de videoconferencia del lado local se cerrará y nos aparecerá de nuevo la ventana de menú principal de Teletrófono. En caso de que la videoconferencia estuviese siendo grabada se finalizará también la grabación.

Ahora debemos considerar el evento que realmente nos interesa: el caso en que el evento *ReceiveStreamEvent* sea del tipo *NewReceiveStreamEvent*. En este caso, obtendremos el flujo de datos recibido y a su vez obtendremos el objeto *DataSource* asociado a éste. Luego obtenemos el objeto de tipo *RTPControl* asociado a esa *DataSource* y a partir de éste tratamos de obtener el formato de los datos recibidos y el nombre canónico del participante que envió el flujo. Algunas de las sentencias que hay que escribir para llevar a cabo estas acciones son:

```
ReceiveStream stream = ((NewReceiveStreamEvent)event).getReceiveStream();  
DataSource dsVideo = stream.getDataSource();  
RTPControl ctl = (RTPControl)dsVideo.getControl("javax.media.rtp.RTPControl");  
Format vf = ctl.getFormat();  
String nombreParticipante = participant.getCNAME();
```

Ya tenemos nuestro flujo de vídeo en forma de *DataSource*, ahora sólo nos faltaría reproducirla por pantalla. Para ello, llamaremos a un método de la clase ‘VideoconfLocal’ que hemos llamado *setPlayer*, al que le debemos pasar como parámetros la fuente de datos de vídeo recibida desde remoto y el formato de ésta:

```
rx.setPlayer(dsVideo, vf);
```

Este método lo que hace en primer lugar es comprobar que la fuente de datos de vídeo que se le pasa como parámetro no sea nula e inmediatamente después crea un *Player*, ya en estado *Realized* pasándole dicha fuente de datos. Luego hay que “conectar” y “comenzar” con la fuente de datos:

```
Player player = javax.media.Manager.createRealizedPlayer(dsVideo);  
dsVideo.connect();
```



```
dsVideo.start();
```

A continuación deberemos extraer el componente visual del Player mediante la sentencia `player.getVisualComponent()`, y añadir dicho componente al panel de la clase `'VideoconfLocal'` en que queramos que aparezca la imagen del profesor en remoto. Por último, pondremos en marcha la reproducción con la sentencia: `player.start()`.

Una vez implementada esta parte de la comunicación, deberemos llamar a otras clases que realicen el resto de funcionalidades. Hemos logrado recibir vídeo y reproducirlo en local, ahora debemos hacer lo mismo con el audio que también recibimos desde el lado remoto. Volviendo entonces a la clase `'ReceptorVÍdeo'`, inmediatamente después de llamar al método `setPlayer`, llamaremos entonces a la clase `'AudioLocal'`. A la clase `'AudioLocal'` le pasaremos como parámetro la variable `'destinoAudio'`, que ya se ha comentado, así como el dispositivo y formato de captura de audio escogido por el usuario. Recordamos que el cometido de esta clase será capturar el audio en el lado local y enviarlo a remoto, así como recibir el audio desde remoto y reproducirlo en local. Como esta clase establecerá también una nueva conexión RTP, necesitaremos recibir notificación de los eventos de este protocolo, por este motivo, esta clase implementa las clases abstractas de Java `'ReceiveStreamListener'` y `'SessionListener'`. Esta clase también implementa la clase `'ControllerListener'`, más adelante explicamos por qué.

En primer lugar deberemos crear una *DataSource* con los datos capturados por el dispositivo escogido por el usuario, que lo hemos almacenado en una variable local del tipo *CaptureDeviceInfo* llamada `'dispositivoAudio'`:

```
DataSource dataSo = Manager.createDataSource(dispositivoAudio.getLocator());
```

Ahora, deberemos establecer formato de audio escogido por el usuario para el dispositivo de captura de audio que también este usuario escogió. Para implementar esto hay que ejecutar una serie de sentencias algo liosas y desde luego nada intuitivas:

```
CaptureDevice dispCaptura = (CaptureDevice) dataSo;
FormatControl [] fcs = dispCaptura.getFormatControls();
(...)
FormatControl fc = fcs[0];
Format [] formats = fc.getSupportedFormats();
for (int i = 0; i < formats.length; i++) {
    if (formats[i].matches(format)) {
        format = formats[i].intersects(format);
        fc.setFormat(format);
        break;
    }
}
```

Tras esto, sólo nos queda establecer la propia conexión RTP para iniciar la transferencia bidireccional de audio entre los dos extremos de la comunicación. Este proceso lo implementamos del mismo modo que lo hicimos para establecer la conexión RTP en la clase `'ReceptorVÍdeo'`. Deberemos crear de nuevo un objeto del tipo *RTPManager* y asociarle los oyentes que implementa nuestra clase. En este caso las direcciones IP de los extremos no varían, pero sí varían los valores de los puertos usados tanto en el lado local como en el remoto.

```
RTPManager mng = RTPManager.newInstance();
mng.addReceiveStreamListener(this);
mng.addSessionListener(this);
mng.initialize( new SessionAddress(ipAddrLocal, PtoAudioLocal1) );
mng.addTarget( new SessionAddress( ipAddrRemota , PtoAudioRemoto1) );
```

Tal y como se explicó en el apartado en el que llevamos a cabo el diseño de la aplicación Teletrófono, deberemos crear un procesador que codifique los datos capturados por el micrófono a un formato entendible por el protocolo RTP. Una vez se halla realizado esta transformación deberemos obtener la salida del *Processor* y proporcionar dichos datos de salida al *SessionManager*. Éste último será el encargado de hacer llegar los datos de interés al otro extremo a través de la red. Se recomienda consultar la figura 4.20. Creamos entonces ese procesador que necesitamos, pasándole como datos de entrada los datos capturados por el micrófono, asociándole un oyente del tipo *ControllerListener*, ahora veremos para qué, y tratando de pasarlo al estado *Configured*:

```
Processor proc = Manager.createProcessor(dataSo);
proc.addControllerListener(este);
proc.configure();
```

El método *controllerUpdate* del oyente *ControllerListener* lleva asociado un evento *ControllerEvent* al que llamaremos ‘event’, así que lo que haremos en primer lugar, será comprobar si ese evento es un caso de un *ConfigureCompleteEvent*. En este caso lo que significa es que el procesador que hemos creado, ya ha sido configurado y, por tanto, ya podremos pasar a definir el tipo de procesamiento que queremos llevar a cabo. Habrá que establecer un formato RTP específico y especificar el descriptor del contenido de salida que deseamos. Recordamos que los formatos se establecen obteniendo el *TrackControl* de cada pista y llamando al método *setFormat* para especificar un formato RTP concreto. El formato de salida lo estableceremos con el método *setContentDescriptor*. Como no requerimos ningún tipo de multiplexación en concreto vamos a fijar este descriptor como “*ContentDescriptor.RAW\_RTP*”. Como códec de audio, se ha utilizado G.723 por ser el que habitualmente se usa para realizar videoconferencias, es el usado, por ejemplo, por el estándar de la I.T.U. H.323. Para realizar estas acciones:

```
proc.setContentDescriptor(new ContentDescriptor(ContentDescriptor.RAW_RTP) );
establecerFormato( proc.getTrackControls(), new AudioFormat(AudioFormat.G723_RTP) );
proc.realize();
```

Siendo el método *establecerFormato* el que sigue:

```
private static void establecerFormato(FormatControl[] fca,AudioFormat format) {
    loop:
        for (int i=0; i<fca.length; i++) {
            Format[] supported = fca[i].getSupportedFormats();
            for (int j=0; j<supported.length; j++) {
                if (format.matches(supported[j])) {
                    fca[i].setFormat(supported[j]);
                    break loop;
                }
            }
            fca[i].setEnabled(false);
        }
}
```

Por el contrario, si el *ControllerEvent* resulta ser un *RealizeCompleteEvent* significará que el procesador ya ha pasado a estado *Realized* y, por tanto, podemos ya obtener los datos de salida de éste y convertir esos datos en un flujo RTP que se envíe al otro extremo de la comunicación. En concreto, en caso de un *RealizeCompleteEvent* ejecutaremos las sentencias:

```
SendStream ss = mng.createSendStream(proc.getDataOutput(), 0);
ss.start();
```

```
proc.start();
```

*\*Nota:* El segundo parámetro que se le pasa al método *createSendStream* es el “índice de flujo”. Si lo fijamos a cero, indica que queremos que todos los flujos de esta fuente de datos sean mezclados en un único flujo de una única fuente (un único SSRC).

Mediante estas últimas sentencias ya estamos logrando enviar el audio que capta nuestro micrófono (del lado local) hacia el lado remoto. Ahora falta recoger el audio que desde remoto se nos está enviando a nosotros para reproducirlo y poderlo escuchar. Esto lo haremos a través del método *update* del oyente *ReceiveStreamListener*. Este método lleva asociado un evento *ReceiveStreamEvent* al que llamaremos ‘event’. El evento del tipo *NewReceiveStreamEvent* es el que más nos interesa. En ese caso obtendremos el flujo recibido mediante la sentencia:

```
ReceiveStream rStream = event.getReceiveStream();
```

E inmediatamente después, llamaremos al método *setPlayer2* de la clase ‘VideoconfLocal’ pasándole este *ReceiveStream*. La función de este método, será reproducir ese flujo de audio que hemos recibido desde remoto. Para ello, tendrá que obtener en primer lugar la fuente de datos de ese flujo. Después crearemos un *Player*, directamente en estado *Realized* como hemos venido haciendo hasta ahora, conectamos la fuente de datos y comenzamos la reproducción:

```
DataSource dsAudio = rStream.getDataSource();  
Player player = javax.media.Manager.createRealizedPlayer(dsAudio);  
dsAudio.connect();  
dsAudio.start();  
player.start();
```

Hasta ahora hemos conseguido recibir el vídeo del profesor en remoto y reproducirlo por pantalla en local, recibir el audio del profesor en remoto y también reproducirlo en local, y enviar nuestro audio al lado remoto. Todo esto si recordamos se lleva a cabo cuando el usuario de la aplicación Teletrófono pulsó el botón de “Iniciar Videoconferencia” y entonces la clase ‘VideoconfLocal’ llamó a su vez a la clase ‘ReceptorVideo’. Bien, pues la clase ‘VideoconfLocal’, inmediatamente después de llamar a la clase ‘ReceptorVideo’, llama a la clase ‘ReceptorVideo2’. Esta clase se encargará de establecer otra comunicación RTP para recibir video desde remoto, establecer otra comunicación RTP para recibir audio desde remoto, y finalmente llamar a las funciones necesarias para almacenar en nuestra máquina el audio y vídeo recibidos en el caso en el que el usuario así lo demande. En definitiva, ‘ReceptorVideo2’ es la clase principal que implementa, junto con ‘AudioLocal2’ como veremos, la herramienta de grabación de videoconferencia que ofrece Teletrófono al usuario que se encuentra en el lado local.

En la figura 4.37 aparece un esquema para visualizar más claramente el orden de llamada de las clases, qué clase llama a cuál y qué tarea realiza cada clase que hemos nombrado para el lado local. Como vemos en dicha figura, la clase que se ejecuta en el lado local es ‘VideoconfLocal’. Esa clase llama a las clases ‘ReceptorVideo’ y ‘ReceptorVideo2’. La clase ‘ReceptorVideo’ se encarga de recibir el vídeo que el extremo remoto envía, pasarle ese vídeo a la clase ‘VideoconfLocal’ para que lo reproduzca y llamar a su vez a la clase ‘AudioLocal’. La clase ‘AudioLocal’ se encarga de recibir el audio que el extremo remoto envía, de enviar el audio que capta el micrófono del lado local y de pasarle el audio recibido a la clase ‘VideoconfLocal’ para que lo reproduzca. Por otro lado, la clase ‘ReceptorVideo2’ se encarga de recibir el vídeo que el extremo remoto envía y llamar a la clase ‘AudioLocal2’ pasándole el flujo

de vídeo recibido. La clase 'AudioLocal2' se encarga de recibir el audio que el extremo remoto envía y de pasarle tanto el audio como el vídeo recibido a la clase 'VideoconfLocal', para que esta última se encargue de almacenar ambos flujos si procede. En el esquema de la figura 4.37 aparecen también indicados los puertos que se utilizarían para cada conexión RTP si el usuario deja en la ventana de configuración los puertos que aparecen por defecto.

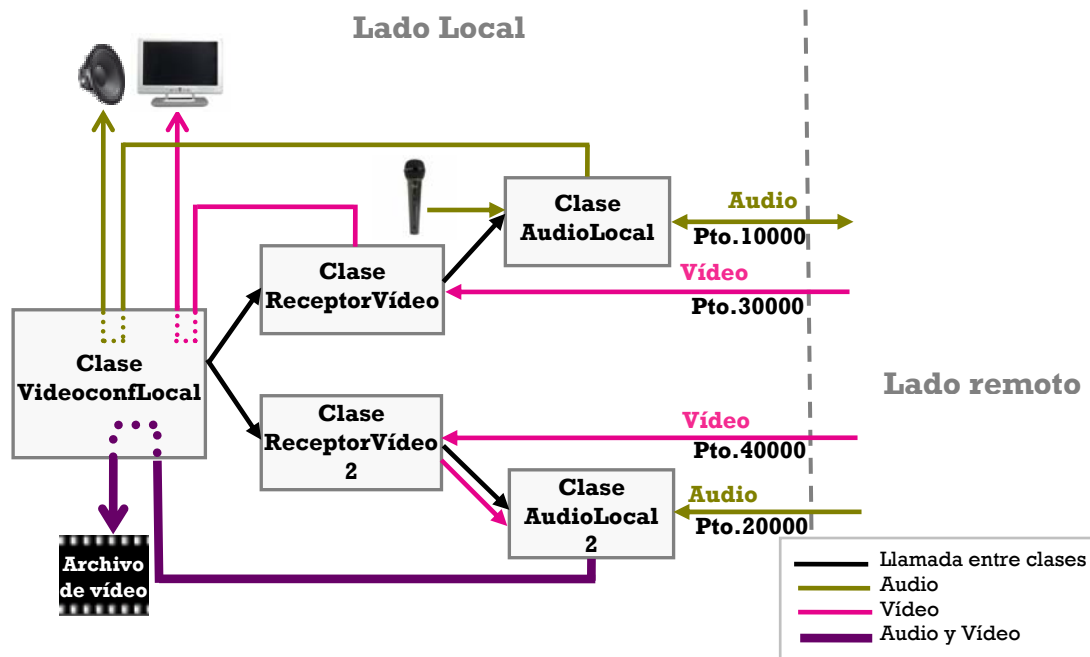


Figura 4.37. Clases en el lado local de la comunicación (I).

Pasamos entonces a ver las clases que nos faltan: 'ReceptorVideo2' y 'AudioLocal2'. La clase 'ReceptorVideo2' será prácticamente idéntica a la clase 'ReceptorVideo', pues su función principal es la misma: recibir el vídeo que el extremo remoto nos está enviando. Debemos, por tanto, establecer una nueva sesión RTP a través de la cual recibir el vídeo del otro extremo. Seguiremos exactamente los mismos pasos para realizar esta tarea que los que realizamos en la clase 'ReceptorVideo'. La única diferencia es que ahora los puertos para esta sesión son otros. Las direcciones IP son las mismas. Ahora:

```
mng.initialize( new SessionAddress(ipAddrLocal, PtoVideoLocal2) );
mng.addTarget( new SessionAddress( ipAddrRemota , PtoVideoRemoto2) );
```

La clase 'ReceptorVideo2' implementará las mismas clases abstractas que las que implementaba la clase 'ReceptorVideo'. La implementación de los métodos *update* de los oyentes de estas clases serán prácticamente idénticos a los explicados para 'ReceptorVideo'. La única diferencia que realmente nos interesa comentar es, que a la hora de implementar el método *update* del oyente *ReceiveStreamListener*, en el caso en que el *ReceiveStreamEvent* sea del tipo *NewReceiveStreamEvent*. En este caso, actuaremos exactamente del mismo modo que hicimos en la clase 'ReceptorVideo' excepto en que: no llamaremos al método *setPlayer* de la clase 'VideoconfLocal' (ni a ningún método de esta clase) y, en vez de invocar al constructor de la clase 'AudioLocal', invocaremos el constructor de la clase 'AudioLocal2', al que, aparte de

pasarle como parámetro la variable ‘destinoAudio2’ (que ya se comentó) y el dispositivo y formato de captura de audio escogido por el usuario, le pasaremos también como parámetro la *DataSource* de vídeo que estamos recibiendo desde el extremo remoto.

La nueva clase ‘AudioLocal2’ será algo más sencilla que ‘AudioLocal’, pues, aunque su función es también recibir el audio que el extremo remoto nos envía, ‘AudioLocal2’ no tiene que enviar al lado remoto el audio capturado por el micrófono situado en el lado local, eso es función exclusiva de la clase “AudioLocal”. Por tanto, tiene que quedar claro que la clase “AudioLocal2” no transmite ningún flujo de datos, tan sólo recibe audio desde el extremo remoto. Para ello hay que establecer igualmente una sesión RTP. La nueva sesión RTP se establecerá de modo idéntico a como lo hicimos en la clase ‘AudioLocal’ sólo que ahora los puertos usados para esta comunicación son otros. Haremos:

```
mng.initialize( new SessionAddress(ipAddrLocal, PtoAudioLocal2) );  
mng.addTarget( new SessionAddress( ipAddrRemota , PtoAudioRemoto2) );
```

Al no tener que enviar ningún tipo de datos a través de la sesión nos ahorramos todo el código que nos servía para acceder al dispositivo de captura de audio y obtener una *DataSource*, así como evitamos tener que crear un procesador, ya que no necesitamos preparar ningún dato para ser transmitido vía RTP. Esto hace que también nos ahorremos, por ejemplo, el método *establecerFormato* que mostramos cuando explicábamos la clase ‘AudioLocal’. La clase ‘AudioLocal2’ implementa entonces las mismas clases que implementaba la clase ‘AudioLocal’ excepto la clase ‘*ControllerListener*’. Ya no necesitamos el oyente que asociábamos al *Processor* porque ya no creamos ningún procesador. Por último, reseñar que en el método *update* del oyente *ReceiveStreamListener*, cuando estamos ante un *ReceiveStreamEvent*, ya no llamamos al método *setPlayer2* de ‘VideoconfLocal’ (pues no queremos reproducir el flujo de audio), pero a cambio, llamamos a un método de ‘VideoconfLocal’ que se llama *configurarGrabación*. A este método le pasaremos como parámetros la *DataSource* de audio recibido (que la obtendremos ejecutando el método *getDataSource* sobre el flujo recibido) y la *DataSource* de vídeo recibida. El citado método se encargará de preparar todos los elementos necesarios para que cuando el usuario pulse el botón “Comenzar grabación” en la pantalla de videoconferencia, se inicie inmediatamente el almacenamiento del audio y vídeo que se le ha pasado como parámetro en un archivo de vídeo.

Olvidándonos por el momento de los *codecs* y contenedores multimedia deseados, tratamos de, tal y como se recomienda en la “JMF API Guide” (que podemos consultar en <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/index.html>), crear, a partir de los flujos de audio y vídeo, una única fuente de datos que será la que trataremos de almacenar. Para crear una fuente de datos que contenga el audio y el vídeo debemos usar el método del *Manager* *createMergingDataSource*, al que le deberemos pasar como parámetro un *array* que contenga las fuentes de datos a mezclar. Hicimos, por tanto:

```
datasourceFinal= javax.media.Manager.createMergingDataSource(new DataSource []  
{dsAudio2, dsVideo2});
```

Sin preocuparnos por el momento demasiado por los *codecs* y formatos, vamos a tratar de almacenar la fuente de datos *datasourceFinal* en un archivo de vídeo ‘.avi’ sin modificar los *codecs* que por defecto traían las fuentes de datos desde el otro extremo (‘G.723’ para la fuente de audio y ‘JPEG\_RTP’ para la de vídeo).

Como ya tenemos las fuentes de datos mezcladas sólo falta prepararlas para que en cuanto el usuario pulse el botón que inicia la grabación, estos datos sean almacenados en un archivo. Para ello, si recordamos los conceptos básicos que vimos de JMF, tendremos que servirnos de un *Processor* que convierta nuestros datos al formato deseado y de un objeto del tipo *DataSink*. Un *DataSink*, como vimos, se usa para leer datos de una *DataSource* y volcarlos en un destino concreto que no sea un dispositivo de presentación.

Como vimos, para construir un *Processor* con ciertas características se puede usar un objeto del tipo *ProcessorModel*. Este objeto, encapsula la información básica requerida para crear un procesador. Al método *Manager.createRealizedProcessor* se le puede pasar como parámetro un *ProcessorModel* para construir un procesador. Vamos a construir un *ProcessorModel* pasándole la fuente de datos fusionada, un *array* de objetos tipo *Format* con el códec deseado para cada pista y un objeto *FileTypeDescriptor* que indica el formato deseado para el archivo de salida. Si le pasamos un *array* de *Formats* que sea *null*, dejará los codecs que por defecto tengan cada una de las pistas. Para el formato ‘.avi’, hay que establecer un *FileTypeDescriptor* con la cadena de texto “video.x\_msvideo”. El código sería entonces:

```
FileTypeDescriptor ftd = new FileTypeDescriptor("video.x_msvideo");
Format[] formats = null;
ProcessorModel pm = new ProcessorModel(datasourceFinal, formats, ftd);
Processor processorGrabación = Manager.createRealizedProcessor(pm);
```

Con esto, ya habríamos terminado la configuración de la grabación, que no empezará hasta que el usuario no pulse el botón de “Comenzar grabación”, presente en la ventana de videoconferencia del lado local.

Supongamos que efectivamente el usuario pulsa el botón de iniciar grabación, en ese momento, pasará a ejecutarse el código de la clase ‘VideoconfLocal’ presente en el oyente de dicho botón. En ese código, se llama a la clase ‘**ChooserGuardar**’, muy parecida a la clase que hemos visto anteriormente llamada ‘ChooserAbrir’. La clase ‘ChooserGuardar’ se encarga de mostrar al usuario un *JFileChooser*, es decir, una ventana a través de la cual puede navegar por los directorios de su ordenador. Mostrará por defecto únicamente los archivos de vídeo del tipo ‘.mov’, ‘.avi’ y ‘.mpeg’, porque internamente llama a la clase ‘FiltroVideo’ que también se ha mencionado anteriormente. El usuario deberá situarse, a través de esta ventana, en el directorio que desee guardar el archivo de vídeo, así como teclear en ella el nombre que le quiera dar a este archivo. No importa que el usuario introduzca el nombre del archivo con su extensión o sin ella porque como hemos dicho, en principio, las grabaciones de van a guardar en ‘.avi’; así que lo que haremos será eliminar siempre la extensión que haya tecleado el usuario (en caso de que haya tecleado la extensión). A continuación, comprobaremos que la ruta que ha introducido el usuario para el archivo de vídeo exista y se pueda escribir en ella y, por último, añadiremos la extensión ‘.avi’ al nombre del archivo. Guardaremos en una variable local llamada ‘miRuta’ una cadena de texto que contiene la ruta y el nombre del archivo de vídeo con su extensión. Por último, llamaremos a un método de la propia clase ‘VideoconfLocal’ que hemos llamado *startRec*.

El método *startRec* en primer lugar obtendrá los datos de salida del *Processor* que creamos. Después, pasará estos datos a un *DataSink*. Para crear el *DataSink* necesitamos también pasarle como parámetro un objeto del tipo *MediaLocator* que contenga la ruta donde el usuario desea guardar el archivo de vídeo. Abriremos el *DataSink* creado y lo

comenzaremos al igual que comenzaremos el *Processor*. En cuanto se ejecuten las siguientes sentencias el vídeo y el audio recibido a través de la videoconferencia desde el extremo remoto comenzará a almacenarse:

```
DataSource outputDS = procesorGrabación.getDataOutput();
MediaLocator ml = new MediaLocator("file://" + miRuta + ".avi");
DataSink datasink = Manager.createDataSink(outputDS, ml);
datasink.open();
datasink.start();
Reloj clock = new Reloj(0,0,0,false);
procesorGrabacion.start();
```

Una vez habíamos implementado la parte de la grabación de la sesión tal y como se acaba de explicar, se pasó a probarla en la práctica para ver si el proceso se ejecutaba correctamente. Efectivamente se almacenaba el audio y vídeo que recibíamos desde remoto en un archivo de vídeo con la extensión ‘.avi’. El archivo se visualizaba y oía perfectamente, sin embargo, el audio y vídeo estaban altamente desincronizados. Más adelante se explicará cómo se resolvió finalmente este problema.

Por el momento hemos logrado grabar en local el audio y vídeo que recibimos desde remoto (aunque el audio y vídeo de la grabación no estén correctamente sincronizados), pero ahora nos falta grabar también los subtítulos.

Como podemos observar en la última porción de código que se ha escrito, antes de comenzar el procesador de la grabación, creamos un objeto del tipo “Reloj” (clase propia que ya se ha explicado). Si un usuario ha pulsado el botón que comienza la grabación, es importante recordar que los subtítulos que a partir de ahora se introduzcan deberán quedar almacenados en un fichero de texto junto con la hora a la que han sido introducidos. A través de este objeto podremos controlar el tiempo en el que el usuario va a introducir cada subtítulo. El reloj comenzará en la hora 00:00:00,000 en el momento en el que comience a grabarse la videoconferencia. Como vimos, el usuario podrá escribir un subtítulo en el panel de inserción de subtítulos. Después deberá pulsar el botón ‘+’ que aparece en este panel. Cuando se pulsa ese botón, el subtítulo aparece bajo la imagen que se recibe desde remoto, tal y como podíamos ver en las figuras 4.35 y 4.36. Pero cuando el usuario pulsa el citado botón, internamente se realizan una serie de operaciones adicionales para lograr almacenar el texto que el usuario ha introducido y el momento en el que lo ha introducido. Estos datos se deben almacenar en un fichero de texto que se llamará igual que el archivo de vídeo y estará ubicado en el mismo directorio. Con esto se pretende que una vez finalice la videoconferencia, el usuario pueda acudir al Reproductor Teletrófono y reproducir el archivo de vídeo y los subtítulos tal y como fueron insertados. Veamos entonces, cómo se implementan estas tareas.

Cuando el usuario situado en el lado local de la videoconferencia pulsa el botón que inserta un nuevo subtítulo, se pasa a ejecutar el código presente en el oyente de ese botón. Ese código, aparte de situar el subtítulo bajo la imagen recibida desde remoto, realiza otras operaciones. En primer lugar deberá obtener la hora que marca el objeto de tipo Reloj que creamos. Luego irá guardando en una variable de tipo *array* (que contiene objetos de tipo *Time* del paquete *sql* de Java) llamada ‘tiempos’, los tiempos en los que se inserta cada subtítulo e irá guardando en otro array (que contiene objetos de tipo *String* de Java) llamado ‘subtítulos’, el texto de cada uno de los subtítulos que se van insertando. Por último incrementaremos una variable de tipo entero que lleva la cuenta del número de subtítulos que el usuario va insertando. Parte del código del oyente del botón ‘+’ será entonces:

```

if (clock != null){
    Time t = new Time(clock.getHoraInt(),clock.getMinutoInt(),clock.getSegundoInt());
    tiempos[contadorSubt] = t;
    subtítulos[contadorSubt] = subtítulo.getText();
    contadorSubt ++;
}

```

Es importante señalar que el anterior código sólo se ejecuta en caso en que el usuario haya optado por grabar la sesión, pues en caso contrario, el objeto ‘*clock*’ será nulo. Como vemos, durante la grabación, simplemente se guardan los subtítulos y sus tiempos en variables locales de la clase ‘*VideoconfLocal*’. Estos datos no se pasarán a un archivo hasta que el usuario no finalice la grabación de la videoconferencia.

Cuando el usuario pulsa el botón de “Parar grabación”, su oyente deberá contener el código necesario para escribir un documento de texto con los subtítulos con el formato que se explicó en la figura 4.26 y además finalizar correctamente el almacenamiento del archivo de vídeo. Por tanto, el oyente, en primer lugar lo que hará será comprobar si el usuario, durante el tiempo que ha durado la grabación, ha introducido algún subtítulo; en caso de que así sea ejecutará el método *escribirDocSubtítulos*. Este método transcribe en un documento de texto los rótulos con su tiempo de inicio y de fin en el formato adecuado. El tiempo de fin de un rótulo se tomará como un segundo antes de que comience el siguiente rótulo. Previamente hemos guardado en los *arrays* ‘subtítulos’ y ‘tiempos’ el texto de los rótulos y el segundo en el que se insertan, respectivamente. La implementación de este método es muy similar a la de otros métodos que vimos para, por ejemplo el Editor Teletrófono. Simplemente consistirá en tener claro cómo es el formato del documento de texto que queremos crear, ir recorriendo los citados *arrays* y escribiendo en un nuevo archivo que nos creamos de tipo *File* a través de un *BufferedWriter* y un *FileWriter*. alguna de las líneas que hay que escribir son:

```

File file = new File (miRuta + ".srt");
BufferedWriter output = new BufferedWriter(new FileWriter(file));
(...)
output.write(texto+"\n");
(...)
output.close();

```

Una vez tenemos escrito nuestro documento con los subtítulos, se deberá parar y cerrar, primero el *Processor* que creamos para realizar la grabación del vídeo, y luego el *Datasink* que usamos para almacenar la grabación. Esto lo haremos a través de los métodos *stop* y *close* de ambos objetos. Pararemos también el objeto Reloj, que ya no necesitamos. En este momento ya tendremos guardados en el directorio que el usuario escogió un archivo de vídeo con la grabación de la imagen y audio que nos llegaba desde el extremo remoto y un documento de texto con los subtítulos que se insertaron mientras se grababa la sesión. Pero el archivo de vídeo lo tenemos guardado en un formato (‘.avi’) y con unos codecs que quizá no son los que realmente nos interesan y además debemos solucionar el problema de sincronización de audio y vídeo que nos aparece a la hora de almacenar ambas fuentes.

Ahora lo que deberemos hacer es escoger el formato y los codecs que realmente queremos utilizar así como tratar a la vez de solucionar el problema de la sincronización del audio y vídeo en la grabación.



### \* Codecs, formatos contenedores y problema de sincronización

Vamos a ver, entonces, los formatos que soporta JMF. Para ello vamos a ver la tabla de formatos que aparece en la página oficial de 'Sun Microsystems' para la versión 2.1.1 de JMF de la que aquí sólo vamos a mostrar de momento, los formatos de vídeo:

Media Type		JMF 2.1.1 Cross Platform Version	JMF 2.1.1 Solaris/Linux	JMF 2.1.1 Windows
AVI (.avi)		read/write	read/write	read/write
Audio	8 bit mono/stereo linear	D,E	D,E	D,E
	16 bit mono/stereo linear	D,E	D,E	D,E
	DVI ADPCM compressed	D,E	D,E	D,E
	G.711 (U-law)	D,E	D,E	D,E
	A-law	D	D	D
	GSM mono	D,E	D,E	D,E
	ACM	-	-	D,E
Video	Cinepak	D	D,E	D
	MJPEG (422)	D	D,E	D,E
	RGB	D,E	D,E	D,E
	YUV	D,E	D,E	D,E
	VCM	-	-	D,E
MPEG-1 Video (.mpg)		-	read only	read only
Multiplexed System stream		-	D	D
Video-only stream		-	D	D
QuickTime (.mov)		read/write	read/write	read/write
Audio	8 bit mono/stereo linear	D,E	D,E	D,E
	16 bit mono/stereo linear	D,E	D,E	D,E
	G.711 (U-law)	D,E	D,E	D,E
	A-law	D	D	D
	GSM mono	D,E	D,E	D,E
	IMA4 ADPCM	D,E	D,E	D,E
	Cinepak	D	D,E	D
Video	H.261	-	D	D
	H.263	D	D,E	D,E
	JPEG (420, 422, 444)	D	D,E	D,E
	RGB	D,E	D,E	D,E

\* D indica que ese formato puede ser decodificado y presentado.  
 \* E indica que el flujo de datos puede ser codificado en ese formato.  
 \* read indica que ese tipo multimedia puede ser usado como entrada (leído de un archivo).  
 \* write indica que ese tipo multimedia puede ser generado como salida (escrito en un archivo)

Figura 4.38. Tabla con los formatos de vídeo soportados por JMF.

Como vemos, no hay mucha variedad donde elegir el tipo de archivo en el que podemos guardar nuestra grabación de la sesión. Vemos que se nos ofrecen, en principio, tres tipos de archivos de vídeo: *Avi*, *Mpeg-1* y *QuickTime*. Pero a nosotros lo que nos interesa es un tipo de archivo en el que guardar nuestro vídeo, es decir, poder escribir un archivo en ese formato. El formato *Mpeg-1* es sólo de lectura, según podemos ver en y la tabla, con lo cual no podemos escribir datos en ese formato. Las posibilidades entonces quedan reducidas a elegir entre dos formatos contenedores de multimedia: *Avi* o *QuickTime*.

Hasta ahora las pruebas realizadas para comprobar si lográbamos guardar el audio y vídeo en un único archivo las hemos realizado guardando en un formato contenedor AVI. Como se ha comentado, el resultado obtenido es un vídeo cuyo audio e imágenes no van correctamente sincronizados. Vamos a tratar de guardar estos flujos en un archivo de tipo *QuickTime* para ver si se mejora la grabación en este aspecto. Si conservamos los *codecs* de audio y vídeo que nos llegan desde remoto, el único cambio importante que habrá que hacer en el código será cambiar el *FileTypeDescriptor* que usábamos para construir el *Processor* ‘processorGrabación’, ahora:

```
FileTypeDescriptor ftd = new FileTypeDescriptor("video.quicktime");
```

Si ejecutamos de nuevo la aplicación, ésta debería guardar sin problemas la grabación en un archivo “.mov”. Pues bien, no siempre ocurre así. La mayor parte de las veces la grabación no se realiza correctamente, almacenándose las imágenes a más fotogramas por segundo de los que debiesen, aunque el audio se almacenaba correctamente. En las pruebas que se llevaron a cabo, las probabilidades de que el vídeo se guardase correctamente o no variaban en función del sistema operativo de la máquina donde se estuviese ejecutando la aplicación Teletrófono. Cuando trabajábamos con un sistema operativo Windows Vista, la grabación se realizaba de modo erróneo (más fotogramas por segundo de los debidos) un 100% de las veces. Mientras que en un sistema operativo Windows XP la grabación se realizaba correctamente aproximadamente un 50% de las veces.

La grabación directa de los flujos en un archivo ‘.mov’ no se conseguía realizar directamente de forma correcta. Tratamos de cambiar la velocidad de los fotogramas antes de guardarlos en el archivo, para bajar los *frames* por segundo, pero no dio resultado. Finalmente consultamos en diversas páginas web de ‘*Sun Microsystems*’ y otros foros acerca de JMF buscando una explicación o una solución a este problema, pero lo único que encontramos fue a personas con el mismo problema que nosotros. La única solución posible para conseguir un archivo de vídeo de tipo *QuickTime* que almacenase la grabación de la sesión de videoconferencia era grabar esta sesión directamente en un archivo ‘.avi’ (como veníamos haciendo hasta ahora) y luego pasar este archivo AVI a uno de tipo *QuickTime* con JMF. Concretamente, se probó a utilizar una clase que proporciona la página web de soluciones JMF de ‘*Sun Microsystems*’ llamada ‘*Transcode*’, que permite convertir un archivo multimedia con unos *codecs* y un formato determinado en otro archivo multimedia con los *codecs* y formato contenedor deseado. En nuestro caso deseábamos que convirtiese nuestro archivo de vídeo ‘.avi’ en un ‘.mov’. Podemos encontrar la citada clase en la página web: <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/Transcode.java>.

Con esta clase que nos proporciona ‘*Sun Microsystems*’ fuimos capaces de almacenar nuestra sesión de videoconferencia en un archivo ‘.mov’ en el que el vídeo se almacenaba finalmente al número correcto de fotogramas por segundo. Sin embargo, seguimos sin solucionar el problema de la sincronización del audio y vídeo. En este caso era de prever que el vídeo de tipo *QuickTime* no iba a solucionar este problema pues hemos tenido que recurrir a una “traducción” de nuestro ‘.avi’, en el que ya existía desfase entre audio y vídeo.

De momento no hemos conseguido solucionar el problema de la sincronización de audio y vídeo en la grabación, pero hemos descubierto la clase ‘*Transcode*’ que nos proporcionan los creadores de JMF, que nos va a servir para jugar un poco con los *codecs* y formatos para aprender algo más de ellos. Pensemos por un momento en los requisitos de nuestro proyecto y qué *codecs* y formato nos gustaría utilizar en función

de éstos requisitos. Lo que a nosotros nos interesa principalmente es que la grabación de la sesión de videoconferencia ocupe el mínimo espacio en disco posible, pensando en que la grabación podrá durar entorno incluso a una hora y que necesitamos poder mover fácilmente ese archivo para que el director de la actividad pueda luego colgar el archivo en una página web, pasárselo fácilmente a sus alumnos o mandarlo por correo electrónico. La calidad de la grabación no nos importa en exceso. Es importante tener en cuenta que la calidad de la videoconferencia no tiene nada que ver con la calidad de la grabación; en la videoconferencia buscamos imágenes grandes y con calidad mientras que en la grabación buscamos ahorrar espacio en disco.

Ahora nos preguntamos ¿qué determina que un archivo multimedia ocupe un mayor o un menor espacio en disco? Pues uno de los elementos que determina fuertemente el tamaño de un archivo es el códec de vídeo usado para codificar en dicho formato. Por ejemplo, un archivo AVI ocupará mayor espacio en disco si el vídeo se codifica en RGB que si se codifica en JPEG. Esto se debe a que el códec JPEG comprime el vídeo, mientras que el RGB no. Sin embargo, el formato del contenedor también determina el tamaño de un archivo. Cada formato contenedor de datos multimedia añade a los datos comprimidos una cabecera (*overhead*); el tamaño de esta cabecera determinará también el tamaño final del archivo.

Centrémonos en primer lugar en la elección de un códec de vídeo. De entre los *códec*s de vídeo que JMF nos permite usar (RGB, YUV, Cinepack, MJPEG, H.261, JPEG, H.263), resulta ser **H.263** aquel que nos proporciona un mayor radio de compresión, aproximadamente de 40:1. El radio de compresión representa la relación entre el tamaño del archivo original y el archivo después de ser comprimido. Este será por tanto el códec que escogeríamos para almacenar el vídeo en la grabación. La calidad de H.263 no es muy buena, pero a pesar de ello se usa mucho habitualmente para la transferencia de vídeo en directo en videoconferencias porque necesita muy poco ancho de banda para llevar a cabo la transmisión. Además H.263 proporciona vídeos únicamente en tres tamaños de imagen distintos: 128x96, 176x144 y 352x288. Nosotros para la videoconferencia, si recordamos, habíamos mencionado anteriormente que el vídeo que recibimos desde el lado remoto de la comunicación viene codificado en **JPEG**. Este códec proporciona un radio de compresión aproximado de 20:1, la mitad de compresión que H.263. Esta elección se llevó a cabo porque implica que durante la videoconferencia veremos un vídeo de mayor calidad que cuando visualicemos la grabación. Además con JPEG existen menos limitaciones en el tamaño de la imagen, tan sólo se debe cumplir que el tamaño de la imagen del vídeo debe ser múltiplo de 8 píxeles. La desventaja de usar durante la videoconferencia el códec JPEG, en vez de H.263, es que necesitaremos hacer uso de un mayor ancho de banda.

Si comparamos ahora los dos formatos contenedores que JMF nos ofrece para guardar nuestro archivo, AVI y *QuickTime*, averiguaremos que la cabecera de un archivo con la extensión ‘.avi’ es mucho mayor que la que se añade en los archivos ‘.mov’. Ello implicará que un archivo tipo AVI ocupará en principio mayor espacio que un archivo ‘.mov’ que contenga los mismos datos que el primero. Atendiendo a esta razón, escogeríamos entonces el formato contenedor *QuickTime*.

En definitiva, guiándonos por la necesidad de que la grabación de la sesión de videoconferencia ocupe poco espacio en disco, decidiríamos que dicha grabación se guardase en un archivo *QuickTime* ‘.mov’ en el que el vídeo vaya codificado en H.263.

Hemos realizado una videoconferencia de prueba de duración un minuto y medio y la hemos grabado en primer lugar en formato ‘.avi’ y con códec JPEG. Después, hemos

traducido el archivo resultante, mediante la clase ‘Transcode’ a: ‘.avi’ con H.263, ‘.mov’ con JPEG y finalmente a ‘.mov’ con H.263. Estos son los resultados obtenidos, ordenados de mayor a menor calidad de imagen del vídeo resultante:

		Espacio en disco		
	AVI (JPEG)	→	495 MB	
	MOV (JPEG)	→	9 MB	
	AVI (H.263)	→	2.8 MB	
	MOV (H.263)	→	1.6 MB	

Como vemos, la calidad del vídeo está directamente relacionada con el tamaño del archivo. A más calidad de vídeo, mayor espacio ocupa el archivo en disco. Efectivamente, tal y como predijimos, el códec JPEG, al tener un ratio de compresión menor que H.263, hará que el archivo ocupe más. Del mismo modo, el formato contenedor, también influye mucho en el tamaño del archivo, vemos que como los ‘.avi’ incluyen una mayor cabecera, ocupan más que los ‘.mov’. Nuestra combinación (Mov + H.263) es la que menos espacio ocupa, pero también la que peor calidad ofrece. Debemos recordar además, que la elección del códec H.263 implicaba la posibilidad de uso de únicamente tres dimensiones de imagen: 128x96, 176x144 y 352x288. La mayoría de cámaras web estándar capturan vídeo con unas dimensiones máximas de 640x480. Estas serán las medidas en las que preferiblemente se capturará y enviará el vídeo del extremo remoto de la comunicación al extremo local. Este vídeo, como ya hemos dicho, se mandará codificado en JPEG, por lo que la calidad será bastante buena. En resumen, durante la videoconferencia nos llegará al lado local un vídeo de grandes dimensiones y buena calidad, pero a la hora de almacenarlo, la cosa cambia: suponiendo que el vídeo se envió de remoto a local con las dimensiones 640x480, si finalmente almacenamos en el archivo mediante H.263 las dimensiones pasarían a ser 352x288. Se reducen las dimensiones del vídeo casi a la mitad. Es el precio que hay que pagar a cambio de ocupar un espacio en disco mínimo.

Hasta ahora lo que hemos hecho es olvidarnos del problema del desfase entre el audio y vídeo de la grabación de la sesión de videoconferencia y centrarnos en qué *codecs* y formato escogeríamos de acuerdo con los requisitos iniciales de la aplicación. De momento la solución parece ser usar: *QuickTime* codificado el video con H.263.

Centrémonos ahora en intentar solucionar el problema de la no sincronización del audio y vídeo de la grabación. Pensemos en qué puede estar pasando...En primer lugar, habrá que comprobar que los flujos de audio y vídeo se estén enviando correctamente sincronizados desde el extremo remoto (aunque aparentemente cuando se lleva a cabo la videoconferencia en directo, no se aprecia retardo alguno entre el vídeo y el audio que se recibe desde remoto<sup>9</sup>. Para ello, en la página web de ‘Sun’ <http://java.sun.com/javase/technologies/desktop/media/jmf/reference/faqs/index.html#jmf2-sync>, en la que se da respuesta a las dudas más frecuentes sobre jmf, encontramos:

“P: ¿Qué debo hacer para sincronizar flujos de audio y vídeo que son transmitidos en sesiones RTP distintas?

JMF soporta sincronización de audio y vídeo sobre RTP(...).Cada flujo de datos RTP apunta a su correspondiente flujo RTCP para determinar correctamente sus ‘timestamps’. Esto se usa para sincronizar los flujos enviados desde el mismo usuario

*identificado por el nombre canónico (CNAME) de ese usuario. Incluso cuando los flujos se envían desde diferentes sesiones (RTPManager), los flujos que provienen del mismo usuario serán sincronizados siempre y cuando todos los flujos lleven el mismo nombre canónico.*

*Así que, desde el punto de vista de la programación, si se crean dos sesiones (RTPManagers) para audio y vídeo, necesitarás asegurarte de que ambas sesiones se inicializan con el mismo 'cname' con el objetivo de garantizar la sincronización"*

De modo que en nuestro caso lo que tenemos que comprobar es que el nombre canónico que uso para cada una de las cuatro sesiones RTP que se establecen sea el mismo. Todas las sesiones se han inicializado con el mismo nombre canónico desde remoto, pero de todas formas, vamos a ejecutar unos métodos sobre cada uno de los cuatro flujos que enviamos. En concreto probamos a imprimir por pantalla los *canonical names* con los que se envían desde remoto el audio y vídeo acudiendo a las clases: 'AudioRemoto', 'AudioRemoto2', 'TransmisorVídeo' y 'TransmisorVídeo2'. En todas ellas pusimos algo similar a:

```
System.out.println("El flujo de audio para reproducir en el otro extremo se envía con el nombre canónico: "+(stream.getParticipant()).getCNAME());
```

obteniendo el mismo nombre canónico para los cuatro flujos, concretamente el nombre canónico es: M<sup>a</sup> Luisa@OrdenadorMary.

Así que tras esta comprobación hemos de suponer que los flujos de audio y vídeo se envían sincronizados desde remoto hasta local. Era de esperar ya que cuando realizamos la videoconferencia, desde el lado local no se aprecia desfase alguno entre el audio y el vídeo que nos envía el profesor en remoto.

Vamos a centrarnos entonces en el lado local. En primer lugar vamos a comprobar si se trata de un desfase entre audio y vídeo lineal, es decir, un mero desplazamiento temporal del audio respecto al vídeo, o por el contrario se produce algún tipo de distorsión no lineal. Para ello vamos a trabajar con un programa de edición de vídeo sencillo, en concreto con "Windows Movie Maker". Vamos a dejar la implementación de nuestro programa tal y como lo creamos inicialmente, es decir, almacenando al audio y vídeo, con los *codecs* que venían por defecto, en un archivo AVI. Vamos a grabar una sesión de videoconferencia Teletrófono de un minuto y veinte segundos de duración. Por último pasamos este vídeo por el "Windows Movie Maker" para tratar de sincronizar vídeo y audio mediante un desplazamiento temporal de este último. Lo que hemos hecho es eliminar el desfase inicial existente entre el audio y el vídeo, sin embargo, tras un minuto veinte segundos de duración del vídeo nos aparece un desfase de dos segundos. Se trata, por tanto, de un desfase entre pistas no lineal, que no podemos eliminar fácilmente. Debemos averiguar qué proceso nos está produciendo ese tipo de distorsión.

Acudimos de nuevo a la "JMF API Guide", donde podemos leer los pasos necesarios para almacenar flujos RTP recibidos desde la red. Comprobamos que los pasos listados en esta guía son los que nosotros hemos seguido al detalle. Tras la explicación de estos pasos hay una nota que indica:

*" Si se desea escribir un archivo que contenga pistas de audio y vídeo, necesitas obtener los flujos de audio y vídeo recibidos de diferentes sesiones y crear una MergingDataSource que contenga ambos flujos. Después pasa la fuente de datos mezclada a Manager.createDataSink."*

Como vemos, tal y como lo hemos hecho nosotros.

Tratando de agotar todas las posibilidades y de encontrar qué proceso o elemento es el que está distorsionando nuestra grabación efectuamos nuevas pruebas. Concretamente, probamos a grabar el audio recibido desde remoto en un archivo y el vídeo en otro archivo independiente, luego probamos a montarlo con “*Windows Movie Maker*” para comprobar si seguía existiendo un desfase no lineal entre ambos. En primer lugar probamos a guardar (con los *codecs* que traen por defecto) el vídeo en un archivo AVI y el audio en un WAV (más adelante veremos los formatos contenedores de audio disponibles en JMF). Había entonces que modificar el método *configurarGrabación* de la clase ‘VideoconfLocal’ para que, en vez de mezclar las fuentes de datos en una *MergedDataSource*, tratase a estas por separado, introduciéndolas en dos *Processors* distintos. Había que modificar también el método *startRec* de la misma clase para que en vez de un *DataSink* crease dos *DataSinks* que nos permitiesen almacenar las salidas de los procesadores en dos archivos distintos. Procedimos entonces del siguiente modo:

```
String outputType1 = "video.x_msvideo";
FileTypeDescriptor ftd1 = new FileTypeDescriptor(outputType1);
Format formats1[] = null;
ProcessorModel pm1 = new ProcessorModel(dsVideo2, formats1, ftd1);
FileTypeDescriptor ftd2 = new FileTypeDescriptor(FileTypeDescriptor.GSM);
Format formats2[] = null;
ProcessorModel pm2 = new ProcessorModel(dsAudio2, formats2, ftd2);
(...)
Processor procesorGrabacion1 = Manager.createRealizedProcessor(pm1);
Processor procesorGrabacion2 = Manager.createRealizedProcessor(pm2);
(...)
DataSource outputDS1 = procesorGrabacion1.getDataOutput();
DataSource outputDS2 = procesorGrabacion2.getDataOutput();
MediaLocator ml1 = new MediaLocator("file:/" + rutaDefinitiva + ".avi");
MediaLocator ml2 = new MediaLocator("file:/" + rutaDefinitiva + ".wav");
DataSink datasink1 = Manager.createDataSink(outputDS1, ml1);
DataSink datasink2 = Manager.createDataSink(outputDS2, ml2);
datasink1.open();
datasink1.start();
datasink2.open();
datasink2.start();
procesorGrabacion1.start();
procesorGrabacion2.start();
```

Con estas líneas, conseguimos que se grabasen por separado el audio y el vídeo en nuestra máquina. Procedimos entonces a montarlos con “*Windows Movie Maker*” para ver si de este modo seguía apareciendo la distorsión no lineal que se producía antes. Resultado: al grabar por separado el audio y el vídeo en distintos archivos, la distorsión se reduce muchísimo (respecto al caso en el que grabábamos las dos fuentes en un único archivo) haciéndose casi inapreciable. Por tanto, podemos deducir que el elemento que estaba distorsionando nuestras fuentes de datos era la *MergedDataSource*.

Sin embargo, no ha terminado aquí nuestra tarea, porque ahora lo que tenemos es un archivo de vídeo ‘.avi’ que nos ocupa mucho espacio en disco y, si recordamos, los requisitos de nuestro programa indicaban que el archivo de grabación de la sesión debe ocupar el mínimo espacio en disco posible. Como ya hemos comentado anteriormente el formato contenedor de *QuickTime* ocupa en principio menor espacio en disco que un AVI. Pero también recordamos que cuando tratábamos de guardar directamente un archivo en ‘.mov’, éste no se guardaba correctamente, almacenándose el vídeo a más fotogramas por segundo de los que debía. Probamos de nuevo a guardar al vídeo en un ‘.mov’, con la esperanza de que la aceleración de los fotogramas fuese otro problema

resultante de utilizar una fuente de datos mezclada. Pero no fue así, al tratar de nuevo de guardar el vídeo en un ‘.mov’, aparece el mismo problema de velocidad de los fotogramas. Teníamos dos opciones:

- usar la clase ‘*Transcode*’ de ‘*Sun Microsystems*’ en nuestra aplicación para que nos “traduzca” nuestro archivo ‘.avi’ que se graba perfectamente en nuestra máquina y sólo contiene vídeo, no audio, a un ‘.mov’;
- optar por otro formato.

Probamos la primera opción, es decir, grabo mi vídeo y audio por separado en archivos ‘.avi’ y ‘.wav’ respectivamente. Cojimos el archivo ‘.avi’ y se lo proporcionamos a la clase ‘*Transcode*’ para que lo convirtiese en un ‘.mov’. Resultado: aumenta de nuevo el retardo y la distorsión entre el audio y el vídeo grabados, por lo que no es una buena solución.

Por lo tanto lo mejor es optar por otro formato contenedor que no nos cree tantos problemas como el de *QuickTime*. Si recordamos la figura 4.38 no tenemos muchas opciones para elegir a la hora de guardar vídeo en JMF, de modo que la única posibilidad que tenemos (una vez desechado el ‘.mov’) es la de guardar el vídeo en un archivo ‘.avi’.

El mayor inconveniente que encontrábamos a los archivos AVI era que ocupaban más espacio en disco que los MOV, sin embargo, la diferencia no es muy grande si tenemos en cuenta que lo que más determina el tamaño de un archivo no es el formato contenedor sino los *codecs* usados para dicho contenedor. En las últimas pruebas que hemos realizado en las que hemos afirmado que el archivo ‘.avi’ obtenido (que contenía el vídeo de la sesión sin el audio) ocupaba gran cantidad de espacio. Esto sucedía porque estábamos utilizando el códec de vídeo que nos viene por defecto desde el extremo remoto: JPEG. Recordando los resultados que obtuvimos realizando pruebas con la clase ‘*Transcode*’:

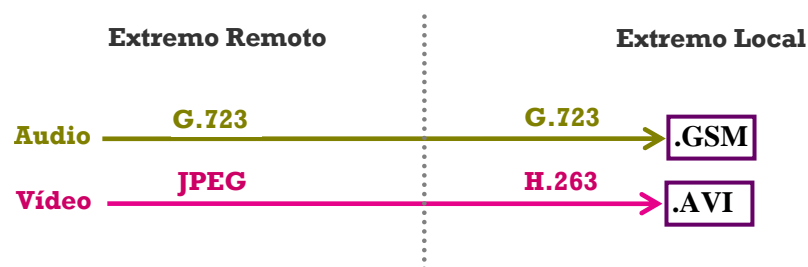
Espacio en disco			
	AVI (JPEG)	→	495 MB
	MOV (JPEG)	→	9 MB
	AVI (H.263)	→	2.8 MB
	MOV (H.263)	→	1.6 MB
			

Vemos que de las cuatro combinaciones que hicimos AVI (JPEG) es la que más espacio ocupa. Lo que tenemos que buscar es un códec de vídeo que unido a un contenedor AVI no ocupe demasiado espacio en disco. Esa combinación es AVI (H.263).

Ya tenemos un archivo de vídeo que no ocupa mucho espacio y que unido a nuestro archivo de audio proporciona un resultado sin apenas distorsión. Ahora nos falta decidir el códec y formato contenedor de audio que vamos a utilizar finalmente. De momento utilizamos un ‘.wav’, que funciona muy bien pero fue una decisión aleatoria no justificada. Veamos los contenedores de audio disponibles en jmf: ‘.aiff’, ‘.gsm’, ‘.mp2’, ‘.au’ y ‘.wav’.

De nuevo nos hemos valido de la clase ‘*Transcode*’ que nos proporciona ‘*Sun Microsystems*’ para llevar a cabo la elección del códec de audio. Lo que hemos hecho es coger la grabación de audio que habíamos realizado en formato ‘.wav’ y la hemos ido pasando al resto de formatos contenedores de audio disponibles (manteniendo el códec de audio G.723, porque debido a su gran radio de compresión no nos interesa cambiarlo). Las conclusiones a las que se ha llegado mediante esta prueba es que los archivos ‘.wav’, ‘.aiff’ y ‘.au’ ocupan el mismo espacio en disco ya que ninguno comprime el audio. Los formatos ‘.gsm’ y ‘.mp2’ sí comprimen el audio, pero este último no he conseguido reproducirlo ni con el “*JMF Player*”. Por tanto, escogeremos como formato contenedor de audio GSM y dejaremos como códec de audio G.723.

Finalmente, los *codecs* y formatos utilizados son:



Tras superar multitud de obstáculos hemos conseguido finalmente poder realizar la grabación de una sesión de videoconferencia, que incluye la grabación del audio que se recibe desde remoto en un ‘.gsm’ (g.723), la grabación del vídeo que se recibe desde remoto en un ‘.avi’ (h.263) y la grabación de los subtítulos que desde este mismo lado (local) insertamos en un ‘.srt’.

Finalmente modificamos de nuevo los métodos *configurarGrabación* y *startRec* de la clase ‘*VideoconfLocal*’ para que finalmente hagan:

```

String outputType1 = "video.x_msvideo";
FileTypeDescriptor ftd1 = new FileTypeDescriptor(outputType1);
Format formatoH263 = new Format(VideoFormat.H263);
Format formats[] = new Format [] {formatoH263};
ProcessorModel pm1 = new ProcessorModel(dsVideo2, formats, ftd1);
FileTypeDescriptor ftd2 = new FileTypeDescriptor(FileTypeDescriptor.GSM);
Format formats2[] = null;
ProcessorModel pm2 = new ProcessorModel(dsAudio2, formats2, ftd2);
(...)
Processor procesorGrabacion1 = Manager.createRealizedProcessor(pm1);
Processor procesorGrabacion2 = Manager.createRealizedProcessor(pm2);
(...)
DataSource outputDS1 = procesorGrabacion1.getDataOutput();
DataSource outputDS2 = procesorGrabacion2.getDataOutput();
MediaLocator m11 = new MediaLocator("file:/" + rutaDefinitiva + ".avi");
MediaLocator m12 = new MediaLocator("file:/" + rutaDefinitiva + ".gsm");
DataSink datasink1 = Manager.createDataSink(outputDS1, m11);
DataSink datasink2 = Manager.createDataSink(outputDS2, m12);
datasink1.open();
datasink1.start();
datasink2.open();
datasink2.start();
procesorGrabacion.start();
  
```



```
procesorGrabacion2.start();
```

Con esto, hemos implementado toda la funcionalidad que se exigía para el lado local de la videoconferencia, quedando finalmente el esquema de clases como se muestra en la siguiente figura:

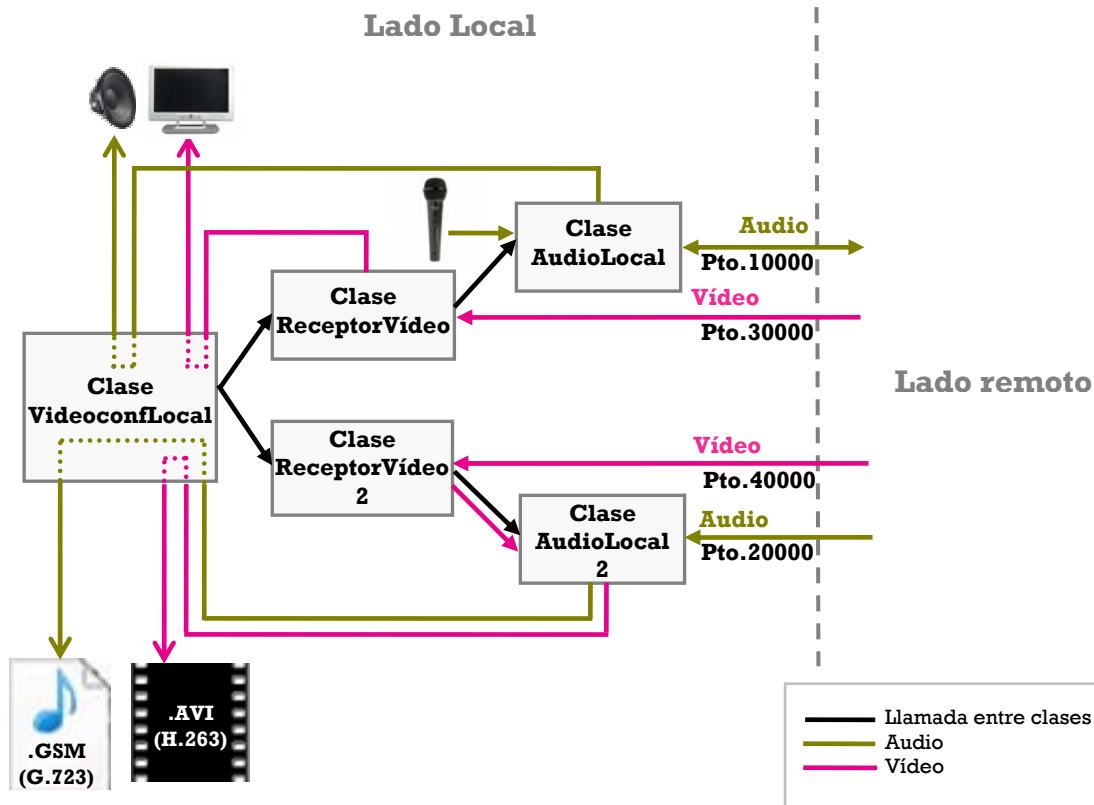


Figura 4.39. Clases en el lado local de la comunicación (II).

## •Extremo Remoto

Ya vimos al inicio del apartado 5.3.3.3 la ventana de configuración de videoconferencia para este extremo de la comunicación. Sabemos, que en cuanto el usuario pulsa el botón que confirma la configuración, a éste le aparece en pantalla una nueva ventana con la que deberá interactuar para comunicarse con el lado local. En la figura 4.40 se puede observar la ventana de inicio de la videoconferencia que aparece en el extremo remoto de la comunicación.

Podemos ver que se trata de una ventana algo más sencilla (con menos elementos) que la del lado local. En la parte central de ésta aparece el mismo letrero que aparecía en local, en el que se indica al usuario el botón que debe pulsar para iniciar la videoconferencia. Bajo éste aparecen, por orden, el botón que nos permite cerrar esta ventana y volver al menú principal de Teletrófono, el botón que debo pulsar para empezar a comunicarme con el lado local y, deshabilitado en este momento, aparece también el botón que nos permite finalizar la videoconferencia. El único elemento que

aparece en esta ventana y que en la ventana de inicio de videoconferencia del lado local no aparecía es un pequeño mensaje situado en la parte superior de la pantalla. En la figura 4.40 apenas se aprecia, pero pone: “----- En este momento su imagen no está siendo transmitida -----”. Es un mensaje de aviso para el profesor que está en este lado de la comunicación, para que sepa que no está transmitiendo todavía datos al otro extremo. Sin embargo, como veremos, cuando la comunicación entre los dos extremos de la videoconferencia se haya establecido, el citado mensaje agrandará su tamaño y cambiará a un color llamativo indicando: “¡ATENCIÓN! Su imagen y voz están siendo transmitidas. Puede dejar de transmitir cuando desee pulsando el botón 'Finalizar Videoconferencia'”. Es un modo de avisar al usuario de que tenga especial cuidado porque ya está transmitiendo su voz y su imagen al otro extremo.



Figura 4.40. Ventana de inicio de videoconferencia del extremo Remoto.

Como se puede observar, en esta pantalla de inicio de la videoconferencia desde el lado remoto no tenemos ni botones para mostrar/ocultar ningún panel, ni ningún botón de inserción de subtítulos, ni botón de grabación, ni nada parecido. Es una ventana sencilla para que sea muy fácil de usar. Esta ventana la implementa la clase ‘VideoconfRemoto’, que será la clase principal que llevará a cabo la videoconferencia en este lado. Tal y como veremos más adelante, desde la clase ‘VideoconfRemoto’ iremos llamando, cuando corresponda, a otras clases de la aplicación que implementarán, cada una de ellas, una porción de la comunicación. Todas juntas, harán posible la videoconferencia.

El usuario en el lado remoto de la comunicación, mediante la videoconferencia Teletrófono, deberá poder verse a sí mismo en su propia pantalla así como mantener una comunicación de audio bidireccional con el lado local de la comunicación. El extremo remoto no tiene disponible ninguna funcionalidad adicional, para que así el profesor en este extremo pueda centrarse en explicar y en mostrar aquello que desee sobre el lugar al que se ha desplazado. Por tanto, cuando el profesor en remoto haya pulsado el botón

“Iniciar Videoconferencia” y se haya establecido la comunicación con el extremo local, la ventana de Teletrófono tendrá un aspecto como el que muestra la figura 4.41. El profesor en remoto, en este caso profesora, se verá a sí misma. De este modo podrá controlar perfectamente y continuamente la imagen que está enviando al otro extremo. Vemos también que aparece sobre la imagen el mencionado letrero que avisa de que la imagen está siendo transmitida. El único botón que está habilitado en el extremo remoto mientras se lleva a cabo la videoconferencia es el botón que permite finalizar la videoconferencia. Como vemos, realizar la videoconferencia desde este lado es muy fácil. La máxima en el diseño del lado remoto fue la sencillez, buscando así la mayor descarga posible de trabajo para el profesor que se encuentre en este extremo.



Figura 4.41. Videoconferencia Teletrófono desde el lado Remoto.

La implementación de este lado de la comunicación será muy parecida a la que se ha llevado a cabo para el extremo local, pero mucho más sencilla, ya que en este lado prescindimos de varias funcionalidades de las que sí está dotado el lado local. En cuanto el usuario pulsa el botón “Iniciar Videoconferencia”, la clase ‘VideoconfRemoto’ crea una instancia de la clase ‘Espere’, que recordamos mostraba una pequeña ventana que mostraba la frase: “Espere... Iniciando la videoconferencia”. La aplicación mostrará esta ventana hasta que la comunicación se haya establecido por completo entre el extremo remoto y el local.

Lo siguiente que hará el oyente del botón “Iniciar Videoconferencia” será, al igual que hacíamos para el lado local, almacenar en variables locales los valores de configuración de la videoconferencia que el usuario fijó, para este extremo, en la ventana de configuración. Recordamos de nuevo acudiendo a las figuras 4.19 y 4.20 que para llevar a cabo la videoconferencia necesitamos hacer uso de cuatro puertos en cada uno de los extremos de la comunicación. Por ello, en la ventana de configuración de la videoconferencia, sea en el lado remoto o en el local, aparecían los valores de los ocho

puertos que se tendrán que utilizar en total. Volviendo a las acciones que lleva a cabo el oyente del botón que inicia la videoconferencia, tenemos que decir que, en concreto, crearemos cuatro variables de tipo *array* preparadas para contener cadenas de texto (*String*). Cada una de estas variables contendrá tres cadenas de texto, como sigue:

- La variable ‘destinoAudio’ contendrá el primer puerto de este extremo para el audio, la dirección IP del lado local y el primer puerto del otro extremo para el audio. Usando solamente los datos que contiene esta variable deberemos ser capaces de establecer una comunicación bidireccional de audio y reproducir en este lado el audio recibido. Para realizar esta tarea nos serviremos principalmente de la clase ‘**AudioRemoto**’.
- La variable ‘destinoVideo’ contendrá el primer puerto de este extremo para el vídeo, la dirección IP del lado local y el primer puerto del otro extremo para el vídeo. Usando solamente los datos que contiene esta variable deberemos ser capaces de capturar nuestra imagen a través de la cámara web, transmitir ese vídeo hacia el lado local y reproducir, en la pantalla del ordenador de este extremo (remoto), la imagen que se está capturando. Para realizar esta tarea nos serviremos principalmente de la clase ‘**TransmisorVideo**’.
- La variable ‘destinoAudio2’ contendrá el segundo puerto de este extremo para el audio, la dirección IP del lado local y el segundo puerto del otro extremo para el audio. Usando solamente los datos que contiene esta variable deberemos ser capaces de transmitir, al lado local, el audio que se captura en este extremo. Para realizar esta tarea nos serviremos principalmente de la clase ‘**AudioRemoto2**’.
- La variable ‘destinoVideo2’ contendrá el segundo puerto de este extremo para el vídeo, la dirección IP del lado local y el segundo puerto del otro extremo para el vídeo. Usando solamente los datos que contiene esta variable deberemos ser capaces de transmitir, al lado local, el vídeo que se captura en este extremo. Para realizar esta tarea nos serviremos principalmente de la clase ‘**TransmisorVideo2**’.

Como vemos las dos primeras variables las usaremos para llevar a cabo la propia comunicación de audio y vídeo que componen la videoconferencia, entre los dos extremos; mientras que las últimas dos variables las necesitaremos para transmitir un flujo adicional de audio, y otro de vídeo, para que el otro extremo sea capaz de almacenarlos si así lo desea el profesor en local.

En la clase ‘VideoconfRemoto’ también se crearán variables locales que contengan el dispositivo de captura de audio que el usuario escogió, el dispositivo de captura de vídeo que el usuario escogió, así como los formatos de audio y vídeo que eligió. Una vez hemos creado todas estas variables con los datos de configuración introducidos por el usuario, el oyente del botón que inicia la videoconferencia, crea una nueva instancia de la clase ‘TransmisorVideo’. Pero antes de meternos de lleno en la descripción de la implementación, veamos la figura 4.42.

En dicha figura se muestra un esquema de las clases utilizadas en este lado de la comunicación y nos permite hacernos una idea general de cómo interactúan entre sí y de su funcionalidad. En el esquema de la figura 4.42 aparecen también indicados los puertos que se utilizarían para cada conexión RTP, si el usuario deja en la ventana de configuración los puertos que aparecen por defecto.

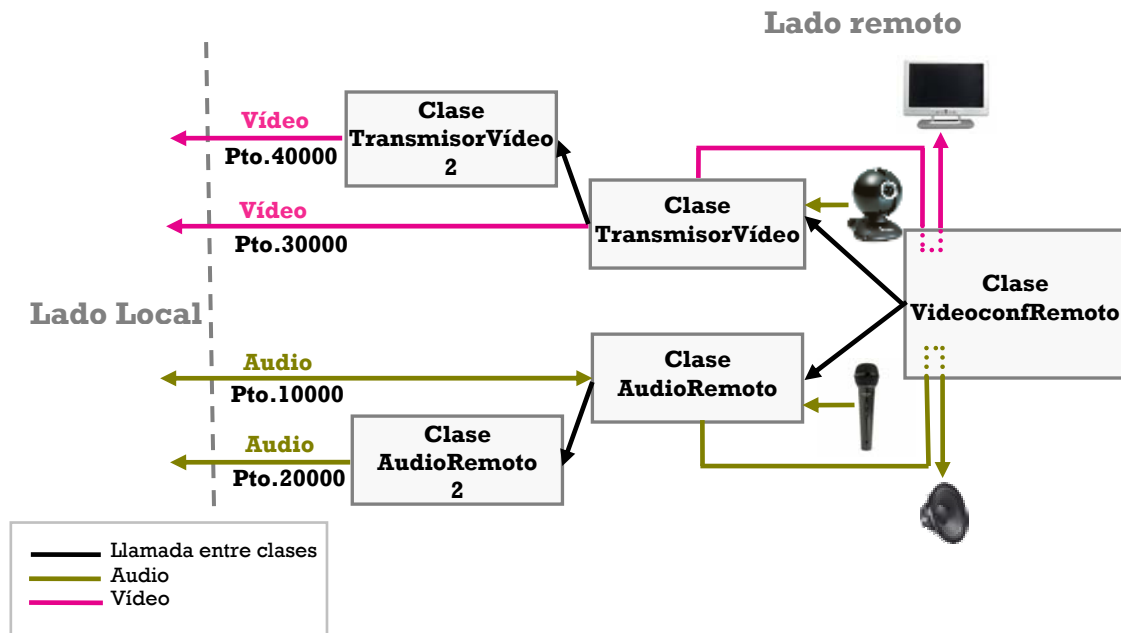


Figura 4.42. Clases en el lado remoto de la comunicación.

Ahora sí, metiéndonos de lleno en la implementación de este lado de la videoconferencia, decíamos que la clase 'VideoconfRemoto' crea una instancia, en primer lugar, de la clase 'TransmisorVideo'. Esta clase se encarga de capturar el vídeo y reproducirlo en este extremo, y enviarlo al lado local para que allí pueda ser visualizado. Esta clase implementa las clases abstractas de Java 'ReceiveStreamListener', 'RemoteListener' y 'ControllerListener'. Su constructor recibe como parámetros, el array 'destinoVideo' y el array 'destinoVideo2' y el dispositivo y formato de captura de vídeo que el usuario escogió. La función principal de esta clase, tal y como hemos mencionado, es capturar vídeo, reproducirlo y establecer una sesión RTP a través de la cual enviar el vídeo al otro extremo. Vamos entonces, en primer lugar, a capturar el vídeo. Suponiendo que tenemos almacenado el dispositivo de captura de vídeo en una variable local de tipo *CaptureDeviceInfo* llamada 'cdi', haremos:

```
String dispositivo = cdi.getName();
MediaLocator locatorDisp = new MediaLocator(dispositivo);
DataSource ds = javax.media.Manager.createDataSource(locator);
```

Ya tenemos la fuente de datos que se captura a través de la cámara web escogida por el usuario. Ahora esta fuente de datos deberemos reproducirla, en este mismo extremo, y enviarla, a través de una sesión RTP, al otro extremo. Para realizar todas estas tareas no podremos usar la misma *DataSource*. Tal y como se explicó en el apartado de diseño, necesitaremos convertir la *DataSource* en una fuente de datos 'clonable', de la que obtendremos, de momento, un clon. Como bien se reflejó en la figura 4.19, la fuente 'clonable' la usaremos para reproducirla en este lado de la comunicación, y, el clon, lo usaremos para transmitir el vídeo al otro extremo. Sin embargo, tal y como vemos en la citada figura (4.19), necesitaremos crear un segundo clon de la fuente de vídeo para pasárselo a la clase 'TransmisorVideo2', que deberá enviar también al lado local el flujo de vídeo para que allí pueda ser almacenado en un archivo. De modo que haremos:

```
dsVideo = javax.media.Manager.createCloneableDataSource(dsVideo);
DataSource dsVideoClon = ((SourceCloneable) dsVideo).createClone();
DataSource dsVideoClon2 = ((SourceCloneable) dsVideo).createClone();
```



Para pasar ahora a reproducir la fuente ‘clonable’, deberemos antes aplicarle el formato de vídeo escogido por el usuario en la ventana de configuración. Para establecer un determinado formato a un dispositivo de captura habrá que obtener los *FormatControls* de éste y ejecutar después el método *setFormat*. El formato deseado lo tenemos almacenado en una variable local de tipo *VideoFormat* que se llama ‘formatoVideo’. Llamaremos entonces al método *establecerFormato*:

```
establecerFormato((CaptureDevice)dsVideo, (Format)formatoVideo);
```

Siendo el método *establecerFormato*:

```
private void establecerFormato(CaptureDevice cdev, Format format) {
    FormatControl [] fcs = cdev.getFormatControls();
    if (fcs.length < 1)
        return;
    FormatControl fc = fcs[0];
    Format [] formats = fc.getSupportedFormats();
    for (int i = 0; i < formats.length; i++) {
        if (formats[i].matches(format)) {
            format = formats[i].intersects(format);
            System.out.println("Estableciendo formato: " + format);
            fc.setFormat(format);
            break;
        }
    }
}
```

Por último, para reproducir la fuente, llamaríamos al método *setPlayer* de la clase ‘VideoconfRemoto’, pasándole la *DataSource* ‘clonable’. Este método es muy parecido al que ya vimos para el lado local de la comunicación. Sus principales sentencias son:

```
Player playerVideo = javax.media.Manager.createRealizedPlayer(dsVideo);
dsVideo.connect();
dsVideo.start();
```

Luego, extraeremos el componente visual del *Player* y lo añadiremos al panel de la ventana que deseemos. Por último ejecutaremos la sentencia:

```
playerVideo.start();
```

Ya hemos logrado capturar el vídeo a través de dispositivo elegido y reproducir la fuente de datos capturada en nuestra propia pantalla del lado remoto. Ahora nos queda transmitir la fuente de datos ‘dsVideoClon’ al otro extremo a través de una sesión RTP. Si recordamos, en el apartado de diseño explicamos que, para transmitir vía RTP un flujo de datos, necesitaremos introducir previamente dicho flujo en un *Processor*. Este procesador será el encargado de convertir los datos de entrada en datos codificados especialmente para ser transmitidos sobre RTP. Haremos:

```
processor = javax.media.Manager.createProcessor(dsClone);
```

A este procesador habrá que añadirle el oyente *ControllerListener* e inducirlo a que pase al estado *Configured*. Una vez está en ese estado pasamos a establecer el formato en el que queremos que salgan los datos del procesador. Recordamos, que para establecer un formato a una pista, debemos obtener el *TrackControl* de ésta y llamar al método *setFormat*. Estableceremos el descriptor de salida como RAW\_RTP, ya que no requerimos ningún tipo de multiplexación en concreto. Esto limitará los formatos soportados reportados por la sentencia “*track.getSupportedFormats()*” a sólo formatos RTP válidos. Es importante mencionar que para formatos de vídeo deberemos

comprobar el tamaño del vídeo, pues no todos los formatos trabajan en todos los tamaños, tal y como vimos cuando explicamos los formatos de vídeo soportados por JMF.

```
TrackControl [] tracks = processor.getTrackControls();
(...)
ContentDescriptor cd = new ContentDescriptor(ContentDescriptor.RAW_RTP);
processor.setContentDescriptor(cd);
Format supported[];
Format chosen;
for (int i = 0; i < tracks.length; i++) {
    Format format = tracks[i].getFormat();
    if (tracks[i].isEnabled()) {
        supported = tracks[i].getSupportedFormats();
        if (supported.length > 0) {
            if (formatoVideo instanceof VideoFormat) {
                chosen = comprobarVideoSizes(format, formatoVideo);
            } else
                chosen = formatoVideo;
            tracks[i].setFormat(chosen);
            System.out.println("Track "+i+" se va a transmitir como:");
            System.out.println(" " + chosen);
        } else
            tracks[i].setEnabled(false);
    } else
        tracks[i].setEnabled(false);
}
```

El método *comprobarVideoSizes* establece el códec de vídeo en el que van a ser enviados los datos. Las opciones que tenemos para codificar el vídeo son:

- JPEG : es una muy buena opción a considerar.
- Cinepack: no es posible codificar datos en este formato con la versión JMF 2.1.1 para Windows, sólo se puede en la versión para Linux. De todos modos este códec nos proporciona muy poca compresión, lo que conllevaría la necesidad de usar para la transmisión del vídeo un gran ancho de banda.
- H.261 : En ninguna de las versiones de JMF 2.1.1 es posible codificar en este formato, tan solo se puede decodificar.
- RGB : No comprime en absoluto el vídeo. Ocuparíamos un ancho de banda excesivo.
- H.263: comprime bastante el vídeo a costa de una pérdida de calidad.

Los principales *códecs* a considerar son entonces JPEG y H.263. Como ya se mencionó, el códec H.263 proporciona una calidad de vídeo no muy buena y suele reducir las dimensiones originales del vídeo, ya que sólo soporta tres dimensiones: 128x96, 176x144 y 352x288. A cambio, dicho formato, permite enviar vídeo consumiendo muy poco ancho de banda en la red. El códec JPEG, por el contrario, proporciona una calidad de vídeo bastante buena y la única limitación de tamaño que exige es que tanto el ancho como el alto de la imagen deben ser múltiplos de 8 píxeles. A cambio los vídeos codificados en JPEG requieren un mayor ancho de banda para su transmisión. Como nosotros deseamos que durante la videoconferencia el lado local reciba una imagen del extremo remoto de grandes dimensiones y con la mayor calidad posible, escogeremos sin duda el códec JPEG. Sin embargo, previniendo cualquier situación, en caso de que el vídeo a transmitir no cumpla el citado requisito de dimensiones que impone JPEG, el vídeo se transmitiría codificado en H.263. Normalmente la aplicación siempre acabará

codificando el vídeo en JPEG, pues las medidas estándar en las que capturan las cámaras web siempre suelen ser múltiplo de 8 píxeles. Vemos entonces cómo actúa el método *comprobarVideoSizes*:

```
private Format comprobarVideoSizes(Format original, Format soportado) {
    int width, height;
    Dimension size = ((VideoFormat)original).getSize();
    Format jpegFmt = new Format(VideoFormat.JPEG_RTP);
    Format h263Fmt = new Format(VideoFormat.H263_RTP);
    if (soportado.matches(jpegFmt)) {
        width = (size.width%8==0 ? size.width:(int)(size.width/8)*8);
        height = (size.height%8==0 ? size.height:(int)(size.height/8)* 8);
    } else if (soportado.matches(h263Fmt)) {
        if (size.width < 128) {
            width = 128;
            height = 96;
        } else if (size.width < 176) {
            width = 176;
            height = 144;
        } else {
            width = 352;
            height = 288;
        }
    } else {
        return soportado;
    }
    return (new VideoFormat(null,new Dimension(width,height),
    Format.NOT_SPECIFIED, null,Format.NOT_SPECIFIED)).intersects(soportado);
}
```

Ahora, pasaremos el *Processor* a estado *Realized* y estableceremos la calidad JPEG a 0.5, que se considera un buen valor por defecto. Guardaremos la salida del procesador del siguiente modo:

```
DataSource dataOutput = processor.getDataOutput();
```

Ahora ya tenemos la fuente de datos de vídeo lista para ser transmitida por una sesión RTP al otro extremo de la comunicación. Seguiremos los mismos pasos que seguimos para establecer cada una de las sesiones RTP en el lado local. En primer lugar, por tanto, crearemos un objeto del tipo *RTPManager* y le asociaremos los oyentes pertinentes:

```
RTPManager mgr = RTPManager.newInstance();
mgr.addReceiveStreamListener(this);
mgr.addRemoteListener(this);
```

Ahora crearemos los objetos *SessionAddress*. Recordamos que la variable 'destinoVídeo' contiene: [Puerto este extremo Vídeo 1, dirección IP otro extremo, Puerto otro extremo Vídeo 1]. Hacemos entonces:

```
SessionAddress origenAddr, destAddr;
int PtoVídeoRemoto1= new Integer(destinoVídeo[0]).intValue();
int PtoVídeoLocal1 = new Integer(destinoVídeo[2]).intValue();
InetAddress ipAddrRemota = InetAddress.getLocalHost();
InetAddress ipAddrLocal = InetAddress.getByName(destinoVídeo[1]);
origenAddr = new SessionAddress( InetAddress.getLocalHost(),PtoVídeoRemoto1);
destAddr = new SessionAddress (ipAddrLocal, PtoVídeoLocal1);
```

Ahora ya podemos iniciar la sesión RTP. Llamamos al método del administrador que inicializa la sesión. Después al método que abre la sesión, al que le pasamos la dirección y puerto del lado remoto, que comenzará a generar informes de control RTCP.



```
mgr.initialize( origenAddr);
mgr.addTarget(destAddr);
```

A continuación comienzo a enviar los datos a través de la sesión creada, poniendo en marcha el procesador:

```
SendStream sendStream = mgr.createSendStream(dataOutput, 0);
sendStream.start();
processor.start();
```

Una vez hemos cumplido con todas las tareas que debía llevar a cabo esta clase (capturar vídeo, reproducirlo en este extremo y enviarlo al otro), llamaremos a la clase 'TransmisorVideo2'.

Antes de comentar la clase 'TransmisorVideo2', decir que en la clase 'TransmisorVideo' habrá que implementar los métodos *update* de los oyentes *ReceiveStreamListener*, *RemoteListener* y *ControllerListener*. El único evento que nos interesa comentar es el evento *ByeEvent*, que será del tipo *ReceiveStreamEvent*. En este caso se mostrará al usuario una ventana, implementada con la clase 'DiálogoEstandar' que ya mencionamos en otras ocasiones, en la que se le informará de que el usuario con el que estaba conectado ha abandonado la sesión. La ventana de videoconferencia del lado local se cerrará, al igual que el procesador y el *Player* que estábamos utilizando. A cambio nos aparecerá la ventana del menú principal de Teletrófono.

Pasamos ya a la clase 'TransmisorVideo2'. Esta clase se encarga de enviar el vídeo capturado en este extremo remoto al otro extremo de la comunicación para que allí pueda ser almacenado en un archivo, si el profesor en el extremo local lo desea. Esta clase recibe como parámetros la variable 'destinoVideo2', el formato de vídeo escogido por el usuario y el segundo clon de la fuente de datos 'dsClone2' que contiene el vídeo capturado.

No vamos a explicar extensamente esta clase porque se trata de una clase idéntica a la clase 'TransmisorVideo' con únicamente un par de diferencias:

- No llamamos al método *setPlayer* de 'VideoconfRemoto' porque no queremos reproducir ningún flujo.
- La sesión RTP que creamos tiene ahora la misma dirección de destino pero no los mismos puertos de comunicación. Ahora en vez de usar para establecer la sesión los datos almacenados en la variable 'destinoVideo', usamos los datos almacenados en la variable 'destinoVideo2':

```
SessionAddress origenAddr, destAddr;
int PtoVideoRemoto1= new Integer(destinoVideo2[0]).intValue();
int PtoVideoLocal1 = new Integer(destinoVideo2[2]).intValue();
InetAddress ipAddrRemota = InetAddress.getLocalHost();
InetAddress ipAddrLocal = InetAddress.getByName(destinoVideo2[1]);
origenAddr = new SessionAddress(InetAddress.getLocalHost(),PtoVideoRemoto1);
destAddr = new SessionAddress (ipAddrLocal, PtoVideoLocal1);
```

Hasta ahora hemos conseguido enviar el vídeo que se captura desde nuestra propia máquina, reproducirlo en nuestra propia pantalla y enviarlo al extremo local para que allí lo visualicen. También hemos conseguido enviar el flujo con el vídeo capturado de nuevo al extremo local para que allí pueda ser almacenado en un archivo. Todo esto si recordamos se lleva a cabo cuando el usuario de la aplicación Teletrófono pulsó el botón de "Iniciar Videoconferencia" y entonces la clase 'VideoconfRemoto' llamó a su vez a la clase 'TransmisorVideo', que a su vez llamó a la clase 'TransmisorVideo2'. Bien, pues la clase 'VideoconfRemoto', inmediatamente después de llamar a la clase

‘TransmisorVideo’, llama a la clase ‘AudioRemoto’ pasándole como parámetros las variables ‘destinoAudio’ y ‘destinoAudio2’, y el dispositivo y formato de captura de audio escogidos por el usuario en la ventana de configuración. Esta clase se encargará de capturar el audio en este extremo, de establecer otra comunicación RTP a través de la cual enviar dicho audio y recibir también el audio que se envía desde local, para pasar a reproducirlo en este lado de la comunicación. En definitiva, es la clase que se encarga de llevar a cabo la comunicación bidireccional de audio entre los dos extremos de la videoconferencia.

La clase ‘AudioRemoto’ es muy similar a la clase ‘AudioLocal’ que ejecutábamos en el otro extremo de la comunicación, pues el cometido de ambas clases es el mismo: enviar y recibir audio. Por tanto, tampoco nos vamos a extender en exceso en la explicación de la implementación de esta clase. Tan sólo vamos a mencionar las diferencias de ésta con respecto a la clase ‘AudioLocal’:

- En ‘AudioRemoto’, cuando creamos la fuente de datos con el flujo capturado desde el dispositivo de audio, deberemos crear una fuente del tipo ‘clonable’, para poder obtener un clon a partir de ella. La fuente ‘clonable’ la usaremos del mismo modo que usamos la fuente de datos de audio en la clase ‘AudioLocal’, es decir, para transmitirla al otro extremo de la videoconferencia en el formato escogido por el usuario. La fuente de datos que es un clon se la pasaremos como parámetro a la clase ‘AudioRemoto2’. Entonces, tal y como hemos dicho, cuando creamos la fuente de datos con el audio capturado deberemos obtener un clon de ésta:

```
DataSource dsAudio = Manager.createDataSource( dispositivoAudio.getLocator() );
dsAudio = javax.media.Manager.createCloneableDataSource(dsAudio);
DataSource dsAudioClon =((SourceCloneable) dsAudio).createClone();
```

- En ‘AudioRemoto’, para reproducir el audio que me llega desde el extremo local, llamaré al método *setPlayer2* de la clase ‘VideoconfRemoto’. Este método es idéntico al método *setPlayer2* de la clase ‘VideoconfLocal’ al que llamaba la clase ‘AudioLocal’.
- Por supuesto, también variarán los parámetros que se utilizarán para crear la sesión RTP. La nueva sesión RTP (llevamos ya tres para este lado de la comunicación) se configurará con los datos que contiene la variable ‘destinoAudio’:

```
SessionAddress origenAddr, destAddr;
int PtoAudioRemoto1 = new Integer(destinoAudio[0]).intValue();
int PtoAudioLocal1 = new Integer(destinoAudio[2]).intValue();
InetAddress ipAddrRemota = InetAddress.getLocalHost();
InetAddress ipAddrLocal = InetAddress.getByName(destinoAudio[1]);
origenAddr = new SessionAddress(InetAddress.getLocalHost(),PtoAudioRemoto1);
destAddr = new SessionAddress (ipAddrLocal, PtoAudioLocal1);
```

- Después de llamar al método *configure* del *Processor* que se utiliza para poner los datos de la fuente ‘dsAudio’ en el formato adecuado para que puedan ser transmitidos por la sesión RTP, crearemos una instancia de la clase *AudioRemoto2* pasándole la variable ‘destinoAudio2’, el formato de audio escogido por el usuario y la fuente de datos clonada, llamada ‘dsAudioClon’, que contiene el flujo de audio que se está capturando en este extremo remoto.

Hemos logrado ya mediante las clases mencionadas capturar vídeo, enviarlo a local para que allí se reproduzca (mediante ‘TransmisorVideo’ con la 1ª sesión RTP), enviarlo a

local para que allí se almacene (mediante ‘TransmisorVÍdeo2’ con la 2ª sesión RTP) y establecer una comunicación bidireccional de audio entre ambos extremos (mediante ‘AudioRemoto’ con la 3ª sesión RTP). Sólo nos queda crear una nueva, y última, sesión RTP para enviar el audio que se captura en el extremo remoto al lado local, para que allí pueda ser almacenado en un archivo junto con el vídeo. Esta funcionalidad la llevará a cabo la clase ‘AudioRemoto2’, a la que acabamos de llamar a través de la clase ‘AudioRemoto’.

A esta clase, como ya hemos dicho, le pasamos la variable ‘destinoAudio2’, con cuyos datos estableceremos esta última sesión RTP, el formato de audio escogido por el usuario y la fuente de audio que tenemos que enviar, ‘dsAudioClon’.

Como siempre, antes de transmitir los datos, deberemos pasar la fuente de datos por un *Processor*. Habrá que establecer un formato RTP específico y especificar el descriptor del contenido de salida que deseamos. Recordamos que los formatos se establecen obteniendo el *TrackControl* de cada pista y llamando al método *setFormat* para especificar un formato RTP concreto. El formato de salida lo estableceremos con el método *setContentDescriptor*. Este proceso lo llevaremos a cabo de forma idéntica a como se hizo en las clases ‘AudioLocal’ y ‘AudioRemoto’. Como no requerimos ningún tipo de multiplexación en concreto vamos a fijar el descriptor como “RAW\_RTP”. Como códec de audio, se ha utilizado G.723 por ser el que habitualmente se usa para realizar videoconferencias.

La RTP para enviar este segundo flujo de audio, se configurará con los datos que contiene la variable ‘destinoAudio2’, de la siguiente manera:

```
SessionAddress origenAddr, destAddr;
int PtoAudioRemoto2 = new Integer(destinoAudio2[0]).intValue();
int PtoAudioLocal2 = new Integer(destinoAudio2[2]).intValue();
InetAddress ipAddrRemota = InetAddress.getLocalHost();
InetAddress ipAddrLocal = InetAddress.getByName(destinoAudio2[1]);
origenAddr = new SessionAddress(InetAddress.getLocalHost(), PtoAudioRemoto2);
destAddr = new SessionAddress(ipAddrLocal, PtoAudioLocal2);
RTPManager mng = RTPManager.newInstance();
mng.initialize(origenAddr);
mng.addTarget(destAddr);
```

La clase ‘AudioLocal2’ implementa la clase abstracta de Java ‘*ControllerListener*’. Como hemos hecho en otras ocasiones, esta clase se añadirá como oyente del *Processor* encargado de adaptar el flujo de audio para enviarlo a través de la red.

Tal y como se explicó para la clase ‘AudioLocal’:

- En el método *ControllerUpdate* de la interfaz *ControllerListener* si se trata de un *ConfigureCompleteEvent*, significará que el *Processor* ha pasado a estado *Configured*, así que será entonces cuando se establecerá el descriptor de contenido y el formato con los que trabajará el procesador (de la forma que ya hemos descrito).
- En el método *ControllerUpdate* de la interfaz *ControllerListener* si se trata de un *RealizeCompleteEvent*, significará que el *Processor* ha pasado a estado *Realized*, así que será entonces cuando, a partir de la fuente de datos de salida del procesador, crearemos el flujo que inmediatamente vamos a enviar con las sentencias:

```
SendStream ss = mng.createSendStream(processor.getDataOutput(), 0);
ss.start();
```

```
processor.start();
```

Con esto hemos finalizado la explicación de la implementación de la herramienta de videoconferencia en el lado local y también la explicación de la implementación de la aplicación Teletrófono.

### 4.3. PROBLEMAS Y CONCLUSIONES.

Finalmente, hemos logrado crear mediante JMF, una aplicación que cumple los principales requisitos que exigimos desde el inicio del proyecto. Hemos conseguido crear un programa que contacta a personas alejadas geográficamente, de forma visual (en un sólo sentido de la comunicación) y auditiva (en ambos sentidos de la comunicación). El diseño de la interfaz con la que debe interactuar el usuario es muy sencilla de usar, por lo que aprender a usar todas las funcionalidades de Teletrófono, a alguien que no tenga una formación técnica específica, no le lleva más que unos minutos. Por este motivo no será necesaria la ayuda de técnicos especializados para poner en marcha la aplicación. Esto da libertad y flexibilidad a los profesores que deseen preparar habitualmente clases haciendo uso de nuestro programa. Hemos logrado asimismo crear una aplicación de videoconferencia versátil que permite al profesor jugar con las herramientas que se le ofrecen de introducción de subtítulos y grabación de la videoconferencia, convirtiéndose éstas en posible fuente de inspiración para realizar otras actividades con los alumnos que aprovechen la sesión de videoconferencia una vez ésta haya finalizado. Aparte de la posibilidad de llevar a cabo videoconferencias, Teletrófono ofrece dos herramientas adicionales que amplían las funcionalidades del programa y aportan valor añadido. Una de estas herramientas adicionales es el Reproductor Teletrófono, que permite reproducir, entre otros, los archivos de vídeo y audio generados mediante la grabación de las sesiones de videoconferencia junto con sus subtítulos; la otra de las herramientas es el Editor de Subtítulos Teletrófono, que permite al usuario modificar, tanto el contenido, como el tiempo de inicio y fin de los subtítulos introducidos por el usuario en una sesión de videoconferencia que haya sido grabada.

Sin embargo, implementar todas estas herramientas y funciones a través de Java y, en concreto, de *Java Media Framework*, no ha sido tarea sencilla. A lo largo del proceso de implementación del programa nos hemos ido encontrando con varios problemas, muchos de los cuales ya han sido mencionados, bien en el apartado de diseño, o bien en el de implementación.

El primero de los problemas que nos encontramos, fue el que en mayor modo nos ha obligado a cambiar el diseño original lógico del programa. Todo comenzó cuando ya habíamos logrado satisfactoriamente llevar a cabo la comunicación propia de la videoconferencia. Es decir, el extremo local de la comunicación recibía y reproducía la voz e imagen que desde el lado remoto se le estaban enviando a través de dos sesiones RTP, y el extremo remoto recibía y reproducía la voz del personaje en local mientras mostraba por pantalla su propia imagen. Hasta ahí todo se pudo implementar con mucho esfuerzo, pero sin ningún problema grave reseñable.

El problema surgió cuando tratamos de realizar la grabación, en el extremo local, del flujo de audio y vídeo que llegaba desde remoto. Lo que tratábamos de hacer, siguiendo las indicaciones de diversos manuales de JMF, era: una vez recibíamos los flujos de audio y vídeo del extremo remoto, los clonábamos en local. Los flujos originales (ahora del tipo '*cloneables*') debíamos usarlos para las funciones que ya estaban implementadas, y que funcionaban correctamente (las propias de la comunicación de la videoconferencia que hemos mencionado), y los flujos clones debíamos usarlos para almacenarlos en un archivo de vídeo, mezclándolos previamente para crear un

*MergedDataSource*. Pues bien, llevar a cabo estas acciones de la forma descrita nos fue totalmente imposible. Podíamos hacer que los flujos RTP que nos llegaban desde remoto fuesen almacenados en un archivo en local, siempre y cuando estos flujos no fuesen reproducidos en local. O bien, podíamos hacer que la comunicación se llevase a cabo de forma impecable, pero sin intentar almacenar los flujos recibidos desde remoto. Había que elegir, reproducir flujos o almacenarlos, a pesar de que estábamos utilizando fuentes de datos distintas, pues habíamos convertido las fuentes de datos recibidas en fuentes ‘clonables’, que usábamos para llevar a cabo la comunicación, y a partir de ellas habíamos creado un clon para cada fuente (pues tenemos una fuente de audio y otra de vídeo). Esos clones eran los que tratábamos de almacenar en un archivo de vídeo. Lo que ocurría exactamente cuando tratábamos de implementar todo a la vez siguiendo este proceso era que la máquina virtual de Java se desplomaba y fallaba, cerrándose la aplicación de forma abrupta. Probamos a trasladar la herramienta de grabación al lado remoto de la comunicación, pero seguía ocurriendo lo mismo, la máquina virtual de Java fallaba.

Como la consulta en manuales JMF y la página de ‘*Sun Microsystems*’ no contemplaba esta situación pasamos a investigar en diversos foros de Internet alguna posible solución. Pero lo que encontramos no fue una solución, sino muchas personas con el mismo problema que nosotros. Es como si para JMF fuese demasiada carga de trabajo capturar audio, recibir un flujo de vídeo a través de RTP, reproducir imagen, recibir un flujo de audio a través de RTP, reproducir audio, enviar audio a través de RTP, clonar las fuentes de audio y vídeo, mezclarlas y almacenarlas en un archivo, todo ello a la vez en el lado local de la comunicación.

Probamos entonces la solución que daría resultado. En la solución que estábamos tratando de implementar, y que no funcionaba, establecíamos únicamente dos sesiones RTP, una para enviar el vídeo del lado remoto al local y otra para enviar en los dos sentidos de la comunicación, audio. La nueva solución, consiste en establecer cuatro sesiones RTP en vez de dos. Una sesión se utilizaría para enviar el vídeo del lado remoto al local y que allí fuese reproducido. Otra sesión se utilizaría para enviar, de nuevo, el vídeo del lado remoto al local y que allí fuese almacenado. Otra sesión se utilizaría para establecer la comunicación bidireccional de audio. Y la última sesión se utilizaría para mandar audio, de nuevo, de remoto a local para que allí fuese almacenado junto al vídeo. De este modo se pueden implementar todas las funcionalidades que deseábamos para la herramienta de videoconferencia, pero a costa de consumir el doble de recursos de red y de seguir un esquema de diseño nada lógico a priori.

Sin embargo, éste no fue el único problema que apareció mientras se llevaba a cabo la fase de implementación del programa. Más allá de la dificultad de aprender y comprender cómo se realizan algunos procesos, nada intuitivos ni sencillos en JMF (como se habrá podido comprobar en el apartado de implementación), surgieron otras dificultades inesperadas.

A la hora de querer almacenar, en el extremo local, el audio y vídeo que recibimos desde el extremo remoto, nos encontramos con pocas opciones de formatos en los que guardar el archivo. JMF apenas ofrece formatos contenedores de vídeo en los que codificar datos. Si recordamos, en la figura 4.38 mostrábamos una tabla en la que aparecían los formatos de archivo de vídeo con los que trabajaba JMF. Son únicamente tres: AVI, MPEG-1 y *QuickTime*. Y teniendo en cuenta que JMF no permite escribir datos en MPEG-1, tan sólo leer este tipo de datos, nos quedan sólo dos formatos contenedores: AVI y *QuickTime*.

La *Java Media Framework* está siendo objeto de muchas críticas por no haber sido capaz de cumplir las expectativas de los programadores. La mayor queja que formulan la mayoría de ellos es que no se han actualizado durante muchos años los *codecs* y formatos con los que puede trabajar JMF, quedando entonces estos obsoletos. A esto se une el nulo trabajo que se ha hecho por mejorar o actualizar este paquete en los últimos años. Este total abandono de JMF por parte de sus creadores provoca que los desarrolladores deban acudir a librerías de terceros de código abierto como la librería *QuickTime* para Java, 'Jffmpeg', 'jvlc', 'FMJ', 'Fobs4JMF', etc.

Uno de los únicos intentos de actualización de los últimos años trató de incluir como códec soportado por JMF, MPEG-4. Sin embargo, los creadores del formato en cuanto se percataron de que JMF era un estándar público, comenzaron a exigir *royalties* por sus derechos de autor. Entonces 'Sun Microsystems' tuvo que eliminar los tres codecs optimizados de MPEG-4, de la distribución de JMF.

Sin embargo, esto que estamos contando no se trata de un problema concreto que nos haya surgido, sino de limitaciones en general que nos encontramos a la hora de desarrollar aplicaciones con JMF. No obstante, sí nos surgió un problema determinado relacionado con este tema de los *codecs* y formatos que vamos a recordar a continuación.

Situémonos de nuevo en el momento de la fase de implementación donde debíamos almacenar, en local, los flujos de audio y vídeo que se reciben desde remoto. En la "*JMF API Guide*" para realizar este proceso se recomienda crear desde un principio una única fuente de datos que contenga las dos pistas (*MergedDataSource*), pasar esta fuente de datos a un único procesador que se encargará de establecer el formato de archivo deseado, e introducir la salida de este procesador en un *DataSink*. Eso mismo fue lo que hicimos, probando inicialmente con un formato AVI. Efectivamente conseguimos almacenar el audio y vídeo en un único archivo '.avi', pero con un desfase no lineal entre el audio y vídeo almacenados. Tratamos entonces de guardar el audio y vídeo en el otro formato contenedor que JMF nos permite, el de *QuickTime*, para ver si el desfase seguía apareciendo. Lo que ocurrió entonces fue que nos percatamos de que al grabar directamente en formato *QuickTime* la grabación de la pista de vídeo se realizaba a más fotogramas por segundo de los que debiese, mientras que la grabación del audio se realizaba correctamente. Acudimos a foros especializados y efectivamente, de nuevo encontramos otros casos como el nuestro en el que al grabar directamente un vídeo en formato *QuickTime*, la imagen se almacenaba al doble de fotogramas por segundo de los que debiese.

Pasamos entonces a asumir que de momento el formato contenedor a usar para almacenar la fuente mezclada del audio y vídeo recibidos desde remoto era AVI. Nos centramos entonces en solucionar el problema de la falta de sincronización entre el audio y vídeo. Aunque durante las sesiones de videoconferencia el audio y vídeo que se recibían en local desde remoto no parecían sufrir ningún retardo o distorsión, debíamos asegurarnos de que desde el lado remoto ambos flujos se estaban enviando sincronizados. Para ello consultamos la página de 'Sun Microsystems' <http://java.sun.com/javase/technologies/desktop/media/jmf/reference/faqs/index.html#jmf2-sync> en la que se indica que para garantizar la sincronización de dos fuentes de audio y vídeo que se envían por dos sesiones RTP distintas simplemente debemos asegurarnos de que ambas sesiones se han inicializado con el mismo nombre canónico. Nos aseguramos de que así lo estábamos haciendo, de modo que la distorsión debía introducirse probablemente una vez que estas fuentes de datos estaban ya en el lado local. En la API de JMF se recomienda para evitar problemas de sincronización justo lo que hemos hecho: crear desde un principio una única fuente de datos que contenga las

dos pistas (*MergedDataSource*), pasar esta fuente de datos a un único procesador que se encargará de establecer el formato de archivo deseado, e introducir la salida de este procesador en un *DataSink*. Siguiendo este proceso, el vídeo y el audio deberían estar sincronizados, pero no es así.

Se nos ocurrió probar a grabar en archivos separados el audio y el vídeo recibidos, prescindiendo de la *MergedDataSource*, luego reproduciríamos ambos archivos a la vez comprobando si persistía el desfase no lineal entre ambos flujos. En concreto, almacenamos el flujo de vídeo en un archivo '.avi' y el de audio en un '.wav'. El resultado: una sincronización del audio y vídeo muchísimo mejor que cuando mezclábamos las fuentes.

Como uno de los requisitos del programa era que la grabación de la sesión de videoconferencia ocupase el mínimo espacio en disco, tratamos de almacenar de nuevo el vídeo en un archivo de tipo *QuickTime* para ver si almacenando en este tipo de archivo únicamente la pista de vídeo (sin la de audio) dejaba de dar problemas. Pero no fue así, de nuevo el vídeo se grababa a más fotogramas por segundo de los que debía. Acudimos a la clase '*Transcode*' que nos proporciona '*Sun Microsystems*' en su página <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/Transcode.java>. Esta clase nos permitió pasar la pista de vídeo que exitosamente habíamos guardado en un archivo '.avi' a un archivo '.mov', sin que aparentemente los fotogramas por segundo se viesen alterados. Sin embargo, cuando reproducíamos a la vez la pista de audio, que habíamos guardado en un archivo '.wav', y la pista de vídeo, que habíamos guardado en un '.mov', volvía a aparecer el retardo y la distorsión de ambas pistas.

En definitiva, si queríamos almacenar el audio y vídeo que se reciben desde remoto durante una sesión de videoconferencia Teletrófono sin sufrir una falta de sincronización entre ambas pistas, debíamos evitar crear una fuente de datos mezclada (*MergedDataSource*) así como evitar usar un formato contenedor para el vídeo de tipo *QuickTime*. No es de extrañar que con JMF tengamos este tipo de problemas con la sincronización de pistas que han pasado a través de una sesión RTP, pues por ejemplo, en una página web de '*Sun Microsystems*' donde éstos han colgado un ejemplo de transmisión de audio y vídeo a través de una sesión RTP, ellos mismos avisan: "*Debido a las limitaciones de JMF 2.2.1, el audio y vídeo no están estrictamente sincronizados.*" (Ver página web:

<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/AVTransmit.html>)

Finalmente tras esta dura lucha con JMF, se implementó una solución definitiva que guarda la pista de vídeo en un archivo AVI codificado en h.263 (para que ocupe el mínimo espacio en disco posible) y guarda la pista de audio en un archivo GSM (para ahorrar también espacio en disco). De este modo la grabación de las sesiones de videoconferencia pueden reproducirse únicamente con nuestro reproductor Teletrófono, pero obteniendo una sincronización mucho mejor entre el audio y el vídeo de la que obtuvimos en todos los intentos anteriores.

Otro problema que nos surgió está relacionado con la memoria de la Máquina Virtual de Java. Cuando llevamos a cabo una videoconferencia, desde el lado local, y la finalizamos, la aplicación nos lleva de nuevo al menú principal de Java. Si realizamos varias videoconferencias seguidas, cuando volvemos al menú principal no se cargan bien las imágenes que lo componen, produciéndose un *java.lang.OutOfMemoryError*. Sin embargo desde el lado remoto la situación es aún peor: cuando iniciamos una segunda o tercera videoconferencia, la webcam deja de capturar, mostrando una imagen en color negro, apareciendo el mismo tipo de error. Esta excepción suele lanzarse



cuando no se están cerrando los *Players* o *Processors* que ya no se usan pero también cuando trabajamos constantemente con imágenes como es nuestro caso.

Hemos de saber que la memoria de la Máquina Virtual se divide en varias regiones. Una de estas regiones es utilizada para, entre otras cosas, guardar el ‘*metadata*’ de las clases como los atributos y sus tipos de datos, métodos etc. Esta memoria es de tipo ‘*non-heap*’. Las instancias de las clases se cargan en la memoria de tipo ‘*heap*’, a la que se van añadiendo y eliminando las instancias de las clases según se van utilizando y eliminándose por el recolector de basura. El tamaño máximo por defecto de la memoria de tipo ‘*heap*’ es de 128 Mb. El sistema operativo presente en la máquina donde se está ejecutando la aplicación determinará fuertemente el comportamiento de la máquina virtual de Java. En ocasiones, y dependiendo de dicho sistema operativo, puede resultar útil, cuando aparecen problemas de este tipo, aumentar la memoria de tipo ‘*heap*’, que es la que a nosotros se nos queda corta; para ello, podemos lanzar el programa de la siguiente manera a través de la línea de comandos:

```
java -Xmx256m Presentación
```

\*El comando `-Xmx256m` establece el tamaño máximo de la memoria de tipo ‘*heap*’ a 256Mb.

Para no tener que escribir esta sentencia en la línea de comandos cada vez que queramos ejecutar la aplicación, a través del Panel de Control de nuestro ordenador, podemos cambiar ese valor de memoria de forma permanentemente. Lo único que hay que hacer es abrir el panel de control del ordenador, hacer doble clic en ‘Java’, en la ventana que nos aparece seleccionar la pestaña Java y pulsar el botón que permite ver la configuración del ‘*Java Runtime Enviroment*’. Aparecerá entonces una nueva ventana que muestra las versiones de Java instaladas; en la fila de aquella versión que nos interese, nos dirigimos a la columna “Parámetros del entorno...” y escribimos ‘`-Xmx256m`’. Por último pulsamos aceptar.

Obviamente la cantidad de memoria que podamos emplear estará limitada por nuestra propia maquina. Según el sistema operativo que estemos utilizando podríamos incluso saturar su memoria y disminuir el rendimiento, por lo que hay que tener precaución si decidimos aumentar la memoria tipo ‘*heap*’. Además el aumento de esta memoria no nos soluciona el problema con la cámara web, que deja de capturar cuando realizamos varias videoconferencias seguidas. Tratando de averiguar qué ocurría nos topamos con una página web de ‘*Sun Microsystems*’ en la que se listan un total de veintitrés problemas que se conocen que tiene la Java Media Framework. La página es: <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/issues.html>. En esta página aparecen varios problemas serios que nos interesan especialmente, a continuación citamos algunos de ellos:

- “*Capturar usando ‘Java Sound’ o ‘Direct Audio’ sólo se puede hacer una vez por instancia de la máquina virtual (JVM). Dependiendo del dispositivo de captura, funcionarán o no, múltiples instancias de ‘Java Sound’ o ‘Direct Audio’ en distintas máquinas virtuales ejecutadas en el mismo ordenador.*”
- “*El paquete JMF para Windows soporta todos los dispositivos de captura que usen la interfaz VFW (Video For Windows). Pero sólo hemos probado JMF con unas cuantas tarjetas y cámaras de captura. Compruebe la lista de dispositivos testados. (...)*”
- “*La captura VFW (Video For Windows) sólo puede usarse una vez por instancia de la máquina virtual.*”
- “*JMF puede quedarse sin memoria en las siguientes situaciones:*

- Si el método 'close' no se llama sobre Players o Processors cuando éstos ya no van a usarse, (...).
- Si hay todavía alguna referencia externa a un Player o Processor que no se esté usando. (...) Una de las menos obvias es que no se estén eliminando los componentes visuales o de control de un reproductor de sus contenedores.
- Algunas grabaciones de vídeo pueden requerir más memoria para reproducirse de la que está disponible en la máquina virtual Java. En ese caso se necesitará ampliar dicha memoria (...).
- La JVM puede también quedarse sin memoria temporalmente después de reproducir sucesivamente varias grabaciones de vídeo."

Como podemos comprobar, los problemas conocidos (del inglés, *known issues*) de JMF que hemos expuesto, son bastante relevantes y graves. Aunque con el paso de los años, nadie de 'Sun Microsystems' parece prestarles demasiada atención. Averiguar que existe esta lista nos ha servido para constatar que el problema que tenemos en "Teletrófono" a la hora de capturar varias veces seguidas con nuestra cámara web, es un problema inherente a JMF y no parece que podamos hacer nada para solucionarlo.

Algunos de los problemas que nos han ido surgiendo nos han obligado a cambiar el diseño y la implementación del programa de modo que, aunque se ha obtenido un programa que funciona y cumple los objetivos planteados en un inicio, por tanto eficaz, no lo hace de una forma eficiente. Pues, por ejemplo, hemos acabado usando el doble de recursos de red que en un principio sería lógico utilizar, enviando dos veces por la red la misma información (hemos acabado usando cuatro conexiones RTP en vez de dos), lo que ha conllevado también que prácticamente hayamos tenido que doblar el código a ejecutar.

En definitiva, JMF nos ha permitido lograr nuestros objetivos casi por completo pero habiéndole dedicado a la tarea mucho tiempo y esfuerzo y debiendo sortear multitud de obstáculos. Es bastante obvio que esta librería necesita actualizarse y comenzar a solucionar, aparte de los '*known issues*', los '*bugs*' (errores) que se le acumulan en la página web '<http://bugs.sun.com/>'. La actualización y mantenimiento de esta librería es también imprescindible para que se renueven los *codecs* y formatos con los que JMF puede trabajar.

Otra de las desventajas de JMF es la poca facilidad que brinda para editar contenidos multimedia. La API de JMF no posee métodos o clases especialmente pensadas para la edición de contenidos. De todos modos sería posible crear aplicaciones que editaran datos multimedia usando modificaciones de la clase DataSource, es decir, customizando esta clase para adecuarla a nuestros propósitos, tarea nada sencilla. En la página web '<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions>' aparecen algunos ejemplos básicos de edición de contenido multimedia en los que deben servirse de *DataSources* customizadas.

El último de los problemas que nos encontramos, que nada tiene que ver con JMF, fue la imposibilidad de conectar a través de Teletrófono dos ordenadores que se encontrasen en redes distintas. Es decir, cuando el extremo local y el extremo remoto se encuentran conectados a una misma red, pueden comunicarse a través de Teletrófono perfectamente. También pueden conectarse perfectamente en el caso en que ambos extremos tengan direcciones IP públicas (direcciones únicas que identifican estos ordenadores de cara al exterior). Sin embargo, si alguno de los extremos está conectado a una determinada red

de área local y el otro extremo está conectado directamente a Internet, o si ambos extremos están conectados a redes de área local distintas, no se hace posible la comunicación a través de nuestra aplicación. Investigando este asunto, nos percatamos de que las direcciones IP que aparecen en nuestra ventana de configuración de videoconferencia (tanto en el extremo local como en el remoto), que se obtienen mediante la llamada en nuestro código a la función `InetAddress.getLocalHost()`, lo que nos devuelve es la dirección IP privada de nuestra máquina. Las direcciones privadas son rangos especiales de direcciones IP que se reservan para ser utilizadas en redes locales y que no pueden ser utilizadas en Internet. Lo que ocurre entonces es que si queremos conectar dos máquinas en redes distintas lo que tenemos que conocer es nuestra dirección IP pública. La dirección pública nos la asigna un NAT (*Network Address Translation*). La idea básica que hay detrás de NAT es traducir nuestra IP privada en una IP pública para que desde nuestra red podamos enviar paquetes al exterior; y traducir luego esa IP pública, de nuevo a la IP privada del ordenador que envió el paquete, para que pueda recibirlo una vez llega la respuesta.

En una típica red casera tenemos un par de PCs que salen a Internet a través del *router*. Cada PC tiene asignada una IP privada, y el *router* tiene su IP privada (puerta de enlace) y su IP pública (que es nuestra IP de Internet).

El mecanismo que utiliza NAT para las asociaciones entre IP pública e IP privada es una tabla (tabla de NAT) en la que guarda una entrada por cada conexión. Cuando un host de la red local inicia una conexión hacia el exterior, el software de NAT asigna una entrada en la tabla, para que a partir de ahora, todo lo que llegue perteneciente a esa conexión sepa traducirlo hacia la IP privada que inició la conexión. Pero si lo que queremos es permitir conexiones desde el exterior a un PC de nuestra red local, hemos de añadir una entrada fija en la tabla de NAT, indicando que todo el tráfico que llegue que vaya a determinado puerto, sea dirigido al PC en cuestión. El puerto es el único elemento que tenemos para “distinguir” conexiones, ya que todo llegará a la IP del *router*, pero tendrán un puerto de destino según sea una conexión u otra. [Fuente: ADSL Ayuda]

Lo que deberíamos hacer entonces si quisiésemos que nuestra aplicación funcionase cuando dos extremos se encuentran en redes de área local distintas (o cuando uno sólo de los extremos se encuentra dentro de una red de área local) es, en primer lugar, conocer las direcciones IP públicas de cada uno de los extremos, y, en segundo lugar, configurar el *router* (o *routers*) para que permitan conexiones desde el exterior a un ordenador de nuestra red de área local.

Para conocer nuestra dirección IP pública a través de nuestro programa Teletrófono podríamos servirnos de STUN (Simple Traversal of UDP over NAT). STUN es un protocolo de red que permite a un cliente detrás de uno o varios NAT determinar su dirección pública y el tipo de NAT que está detrás. Para añadir esta funcionalidad a nuestro programa podríamos servirnos de JSTUN (Java STUN). JSTUN es una implementación basada en Java de STUN. Para ejecutar esta implementación de STUN es necesario bajarse de la página web <http://jstun.javawi.de/> tres archivos ‘.jar’ que contienen el código de esta aplicación. Para conocer nuestra IP pública habría que llamar a una de las clases que contiene uno de estos archivos.

Por tanto, para conseguir que Teletrófono funcionase entre distintas redes, deberíamos cambiar la llamada al método `InetAddress.getLocalHost()` que se hace en las ventanas de configuración de la videoconferencia para ambos lados de la comunicación, por una llamada al método de JSTUN que nos devuelve la IP pública de nuestra máquina.

Además de esto, como hemos dicho, habría que configurar adecuadamente el *router* que hace de puerta a nuestra red de área local para que permitiese la conexión por determinados puertos desde el exterior hasta nuestro ordenador.

Sabiendo ya las tareas a realizar para conseguir que Teletrófono permita la conexión entre dos extremos que se encuentren en redes diferentes cuando al menos uno de ellos tiene una dirección IP privada, se ha considerado que esta funcionalidad excede los objetivos principales planteados para la aplicación y que serán implementados en futuras versiones de Teletrófono. Este proyecto entonces, con Teletrófono, da solución a la comunicación entre: dos ordenadores que se encuentren situados en una misma red, o dos ordenadores que se encuentren situados en redes diferentes pero tengan direcciones IP públicas (no se encuentren detrás de un NAT). Se recalca entonces la conveniencia de WiMAX para la implementación de este proyecto; la gran cobertura de las antenas de este tipo permitirían usar Teletrófono para comunicar dos ordenadores en una misma ciudad, suponiendo que toda la ciudad (por supuesto, no hablamos de una gran ciudad) estuviese cubierta por una sola antena WiMAX.



## **CAPÍTULO V**

### **DISPOSITIVOS A UTILIZAR**

En este capítulo vamos a definir los requisitos y características que han de tener los dispositivos que se usen para llevar a cabo la videoconferencia educativa protagonista de este proyecto. Una vez más, antes de comenzar con la elección, deberemos tener en cuenta desde el principio, la división del entorno planteado, en dos sub-espacios: el espacio del aula en el que se encuentra un profesor con sus alumnos (local), y el espacio remoto en el que está situado únicamente un profesor para explicarles a los alumnos del aula algo sobre dicho entorno.

Nos centraremos principalmente en los dispositivos a usar en el lado remoto de la comunicación. Pues, en el lado local, se plantea en realidad un entorno tradicional de acceso a Internet desde un lugar fijo; situación y soluciones por todos conocidas, pues se puede resolver tal y como resolvemos el modo de acceder a Internet desde nuestros hogares. Por tanto, pasaremos directamente a mencionar los tipos de dispositivos que necesitamos para realizar una videoconferencia que resulte útil y cómoda tanto para los alumnos como para el profesor.

Los dispositivos a utilizar para llevar a cabo la videoconferencia educativa, desde el lado remoto, deben cumplir varios requisitos. En este lado de la comunicación vamos a ser, por tanto más exigentes con los dispositivos. Necesitamos dispositivos móviles, necesitamos que el equipo a través del cual se va a acceder a Internet pueda comunicarse a través de WiMAX (bien que el dispositivo venga directamente preparado para esta tecnología o bien que permita que se le añada algún otro tipo de dispositivo existente que convierta el equipo en WiMAX) y necesitamos que, de acuerdo con los objetivos planteados inicialmente para el proyecto, estos dispositivos no conlleven un alto coste.

## 5.1. DISPOSITIVOS EN EL EXTREMO LOCAL DE LA VIDEOCONFERENCIA.

El lado local de la comunicación es un lugar determinado, fijo. Normalmente se tratará de un aula de un colegio o centro de estudios. No entra dentro de los objetivos de este proyecto dar una solución a este lado para acceder a Internet, principalmente porque es un problema más que resuelto hoy en día y al que prácticamente la mayoría de la población de los países desarrollados se ha hecho frente alguna vez al tratar de dotar a su propio hogar de acceso a la Red. Las posibilidades son múltiples, así que vamos a coger un ejemplo que se nos proporciona desde la página web del Ministerio de Educación, acerca de la instalación que suele haber en los centros de Educación Secundaria de la Comunidad de Madrid. Así nos haremos una idea más aproximada de la infraestructura existente en el lado local de la videoconferencia.

La Comunidad de Madrid provee a los centros de Educación Secundaria de diferentes tipos de instalaciones, pero el caso concreto en el que nos vamos a centrar nosotros habrá dos redes de área local y dos accesos a Internet diferenciados. En primer lugar tenemos la red ICM (Informática Comunidad de Madrid), que utilizan la secretaría del centro, los órganos de dirección y la sala de profesores. Esta red la mantiene por completo el personal de ICM, y el acceso a Internet se realiza a través de un determinado *proxy* (una única dirección IP). La segunda red es utilizada por las aulas de informática. En este centro concreto hay cinco aulas distribuidas en las tres plantas del instituto, con unos 18 equipos cada una. Estos equipos son ordenadores personales y servidores. Cada aula está dotada con un *switch* de 10/100 Mbps al que están conectados todos los equipos. Además en esta página nos cuentan que se ha instalado un filtro utilizando un servidor *proxy Squid* (actúa como un agente, aceptando peticiones de clientes y las envía a los servidores de Internet; cuando el servidor de Internet proporciona la web a nuestro *proxy*, éste se guarda una copia y sirve la información al ordenador que ha hecho la petición) a través del que pasan los ordenadores de estas aulas, con el que se controlan las páginas que visitan los alumnos; para ello utilizan herramientas adicionales que no vamos a comentar. Por último existirá un *router* a través del cual accedemos a Internet, previo contrato con un proveedor de servicios de Internet (ISP).

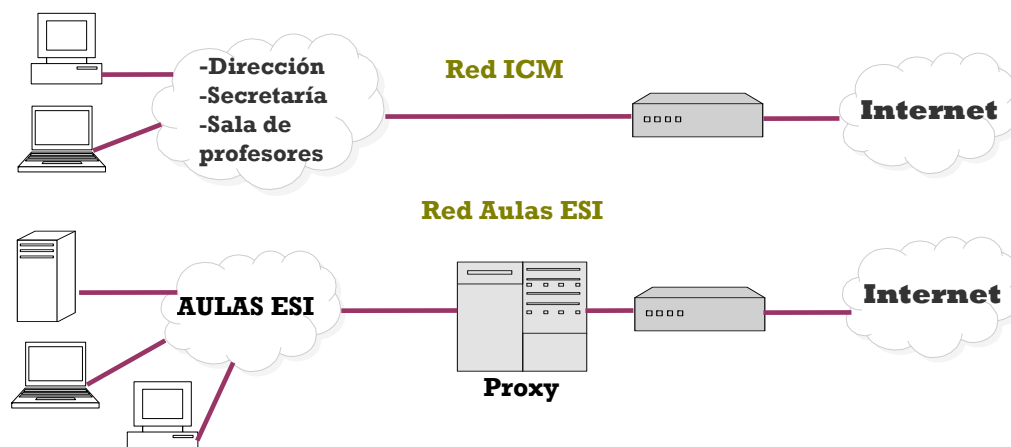


Figura 5.1. Instalación de Red en un Instituto de Educación Secundaria.

En la figura 5.1 se puede ver un esquema general de la red de acceso a Internet implementada en el Instituto de Educación Secundaria que pone como ejemplo el Ministerio de Educación.

Una vez descrito cómo se lleva a cabo el acceso a Internet desde el lado local, nos centramos en los dispositivos necesarios para llevar a cabo la videoconferencia desde este extremo. En primer lugar necesitaremos un ordenador, nos es indiferente que sea de sobremesa o portátil, para ejecutar en él nuestro software de videoconferencia Teletrófono. A través de este software llevaremos a cabo una videoconferencia y, tal y como hemos visto en el capítulo anterior, desde el lado local de la comunicación, podremos tanto enviar como recibir audio. Por tanto, necesitaremos unos altavoces que reproduzcan el audio que se recibe desde el extremo remoto y un micrófono que capture el audio que queramos enviar a ese extremo. En principio nos es indiferente que estos dispositivos de captura y reproducción del audio estén integrados en el ordenador o no, pero sí sería deseable que ambos dispositivos proporcionen una captura/reproducción de calidad. Para el caso concreto de los altavoces, en el supuesto de que el aula no tenga preinstalado un sistema de sonorización, el sonido que salga por ellos, que será la voz del profesor en remoto, deberá llegar a todo el aula, por lo que deberán tener una potencia suficiente para ello. Para que también todos los alumnos presentes en el aula puedan visualizar la pantalla del ordenador, en la que aparecerá la imagen del profesor en remoto y los subtítulos que va introduciendo el profesor que los acompaña en el aula, habrá que hacer uso de un proyector. El proyector permitirá que los alumnos puedan visualizar a gran tamaño la misma pantalla de Teletrófono que está visualizando el profesor que los acompaña.

Como, vemos, todos los dispositivos a usar en el extremo local de la videoconferencia suelen estar disponibles en cualquier centro de estudios, por lo que no habrá que hacer inversión alguna en estos. Tan sólo llevar a cabo la instalación de Teletrófono en el ordenador que se disponga, conectar el proyector, el micrófono y los altavoces, y listo.



## **5.2. DISPOSITIVOS EN EL EXTREMO REMOTO DE LA VIDEOCONFERENCIA.**

En el lado remoto de la comunicación, los dispositivos a usar no están tan claros. Si recordamos, lo que pretendemos, es que este lado pueda estar situado en prácticamente cualquier lugar, tanto interior como al aire libre. La tecnología de acceso a Internet escogida para este extremo es WiMAX, con lo que hemos de suponer que en un futuro, existirá una cobertura total WiMAX a nivel nacional, que nos permitiría acceder a la Red desde cualquier punto que imaginemos.

Como vemos, la situación en este extremo es bastante más específica y exigente que en el extremo local. Necesitamos usar un dispositivo que nos permita ejecutar en él nuestro programa Teletrófono y acceder a Internet a través de WiMAX. Además, necesitaremos altavoces y micrófono, para poder llevar a cabo la comunicación bidireccional de audio, y una cámara web, para grabar la imagen del profesor que está en este extremo. Estos tres dispositivos podrán ir incorporados en el dispositivo principal o simplemente conectarse a éste.

Necesitamos, por tanto, en primer lugar, un dispositivo que sea móvil y permita acceder a Internet a través de WiMAX (bien porque esté especialmente diseñado para trabajar con esta tecnología o bien porque se le pueda añadir algún otro dispositivo que se lo permita). Pero ¿qué quiere decir que un dispositivo sea móvil?, pues quiere decir que tiene un tamaño lo suficientemente pequeño como para llevarlo con uno mismo, que tiene algunas capacidades de procesamiento, que tiene capacidad para conectarse permanentemente o intermitentemente a algún tipo de red de forma inalámbrica y que tiene gran autonomía. Algunos ejemplos de dispositivos de este tipo serían los teléfonos móviles, las PDAs (*Personal Digital Assistant*), los ordenadores portátiles, los *tablet PCs* o los ahora tan de moda *netbooks*.

Descartamos teléfonos móviles y PDAs tan sólo porque si eligiésemos alguno de estos dispositivos deberíamos hacer una adaptación completa de la aplicación Teletrófono a este tipo de entornos. Pero se propondrá como trabajo futuro, ya que el uso de estos dispositivos sería muy interesante principalmente por su reducido tamaño, que harían la labor del profesor en remoto mucho más cómoda.

Por tanto, vamos a considerar principalmente los otros tres tipos de dispositivos que hemos mencionado: ordenador portátil, *tablet PC* y *netbook*. Los tres dispositivos son muy parecidos, tan sólo vamos a mencionar brevemente las características que los diferencian. Los *netbooks* se diferencian de los ordenadores portátiles por su menor peso, tamaño, prestaciones y precio, y una mayor autonomía. Orientados principalmente para el acceso inalámbrico a Internet nunca superan las 13 pulgadas ni los dos kilos de peso. Un *tablet PC*, sin embargo, está a medio camino entre un ordenador portátil y una PDA. En un *Tablet PC* podemos escribir a través de una pantalla táctil, pudiendo el usuario utilizar un lápiz para trabajar con el ordenador sin necesidad de teclado o ratón. Existen modelos que sólo aportan la pantalla táctil a modo de pizarra, pero también hay otros llamados “convertibles” que permiten rotar la pantalla y colocarla como si de una pizarra se tratase.

Ya tenemos en mente los tipos de dispositivos móviles que nos pueden servir para solucionar el soporte de nuestra aplicación Teletrófono. Ahora veamos si es posible que este tipo de equipos sean compatibles con un acceso WiMAX.

### 5.2.1. DISPOSITIVOS MÓVILES CON TECNOLOGÍA WiMAX .

Existe una cantidad considerable de teléfonos móviles y PDAs con WiMAX integrado, sin embargo en esta sección nos vamos a centrar únicamente en los dispositivos WiMAX que nos serían de utilidad en nuestro proyecto.

Como ejemplo de *netbook* con WiMAX integrado tenemos el “Everex CloudBook MAX”. Este *netbook* se puede conectar fácilmente a las redes inalámbricas de banda ancha 802.16e, pero también a las redes 802.11b/g (Wi-Fi). Este dispositivo tiene una pantalla de 8.9 pulgadas y pesa menos de 1 kg. Además incluye una webcam de 2 Megapíxeles. Su batería dura algo más de cuatro horas.

Otro ejemplo de *netbook* con WiMAX sería el “Asus Eee PC 1008HA”, de diseño más moderno que el anterior y con una pantalla LCD de 10 pulgadas. El peso es similar al “Everex”. Este *netbook* de ‘Asus’ viene equipado con 3G y WiMAX. En la figura 5.2 podemos ver el aspecto de este modelo.

Sin embargo, hay otras marcas que también se han lanzado a fabricar modelos de *netbooks* con WiMAX incorporado. Algunos de los modelos existentes son: “Acer Aspire 2930”, “Acer Aspire One AO532h-Bk86X”, “Samsung NC10”, “Onkyo C204A5”, “Sony VAIO P WiMAX Edition”, etc. Sin embargo, algunas marcas están llevando a cabo la comercialización de algunos de éstos modelos únicamente en Corea (donde WiMAX está teniendo gran éxito) o en Japón.



Figura 5.2. “Asus Eee PC 1008HA”. Un *netbook* con WiMAX.

Pero la industria no se ha centrado únicamente en integrar la tecnología WiMAX en los exitosos *netbooks*, también lo ha hecho en los tradicionales ordenadores portátiles. Como por ejemplo en el “Acer Aspire 5930”, compatible también con la tecnología 802.11n, o en la serie “Acer Aspire Timeline”. También WiMAX está ya presente en *notebooks* de otras marcas, como se puede comprobar con los modelos “Lenovo ThinkPad X301”, “Toshiba Portégé R600-STS520W”, “Dell Studio 15”, “Dell Studio 16” y “Dell Studio XPS 16” o “Fujitsu LifeBook P8020”.

Sin embargo, hay novedosos dispositivos, como del que vamos a hablar a continuación, que también incorpora tecnología WiMAX y que podrían sernos muy útil para llevar a cabo el proyecto de la videoconferencia educativa. El dispositivo en cuestión se llama “Samsung SPH-9200” y se trata de un dispositivo ultra portátil mezcla entre un *tablet* PC, una PDA y un notebook. Este dispositivo fue presentado en 2007 en Corea, pero parece que por el momento, no va a llegar a Estados Unidos ni Europa, y no será por falta de admiradores en estos lugares.

El “Samsung SPH-9200” es extensible, posee una pequeña pantalla táctil LCD de 800x480 puntos de resolución y 5 pulgadas tiene un disco duro de 30 GB, lleva incorporada una webcam de 1,3 Megapíxeles, tiene una autonomía de dos horas y utiliza un sistema operativo “Windows Xp Home Edition”. Además cuenta con puertos Ethernet LAN, varios puertos USB, ranura para tarjetas SIM, etc. Además al extenderlo, cuenta con un completo teclado QWERTY (la distribución de teclado tradicional). Permite conexiones inalámbricas Wi-Fi, WiMAX (WiBro) y HSDPA. En la figura 5.3 podemos ver el aspecto de este dispositivo, que se puede plegar en diferentes posiciones. La primera de las imágenes que aparece en la figura es el dispositivo “Samsung SPH-9200” cuando abrimos la pantalla, luego cuando abrimos el teclado y por último se muestra a su antecesor, el “SPH-9000”, para mostrar las multitudes de posiciones que puede adoptar (en ese aspecto ambos modelos son idénticos).



Figura 5.3. “Samsung SPH-9200”. Un dispositivo ultra ligero con WiMAX.

Pero, ¿qué ocurre si queremos seguir usando nuestro ordenador portátil, *netbook* o *tablet* PC que no llevan incorporado WiMAX? Para eso han salido al mercado pequeños dispositivos que podemos conectar a nuestro equipo de trabajo habitual para convertirlo en un dispositivo WiMAX.

Los primeros dispositivos de este tipo que aparecieron fueron del tipo tarjeta PCMCIA como la “SWT-P230” que, de la mano de ‘Samsung’, nos ofrecía conectividad WiMAX

802.16e. Al operar en la banda de 2,3 GHz, ya podemos intuir que no es un dispositivo para el mercado europeo. En la figura 5.4 podemos ver el aspecto de esta tarjeta.



Figura 5.4. “SWT-P230”. Tarjeta PCMCIA para conexión WiMAX en 2,3 GHz.

Pero ahora también hay disponibles otros tipos de dispositivos para adaptar nuestro ordenador a WiMAX, como por ejemplo el producto “MiMAX USB” del fabricante ‘Airspan’. Se trata de un dispositivo USB cuatri-banda que convierte cualquier dispositivo en un dispositivo listo para usar WiMAX.

“MiMAX USB” es lo que se conoce como un dispositivo ‘*plug-and-play*’ (enchufar y listo) y es compatible tanto con sistemas operativos ‘Mac’ como con ‘Windows’. Para posibilitar el *roaming* global, “MiMAX USB” trabaja con cuatro bandas distintas de frecuencia: 2.3-2.4 GHz, 2.5-2.69 GHz, 3.3-3.8 GHz y 4.9-5.8 GHz.

Se trata de un dispositivo que a pesar de sus dimensiones posee una antena de grandes funcionalidades con tecnología MIMO y que nos permite alcanzar velocidades de más de 33Mbps (en un canal de 10 MHz). Además se ha diseñado de modo que sea muy eficiente en cuanto al uso de la potencia del dispositivo al que se ha conectado.

Pero el MiMAX USB es sólo el primer producto de una familia de dispositivos móviles WiMAX que ‘Airspan’ quiere lanzar. De momento, ha lanzado ya otro dispositivo de esta familia: el ‘*MiMAX Finder*’ (buscador MiMAX).

Se trata de un accesorio para el ‘MiMAX USB’ que al insertarle éste, detecta redes WiMAX disponibles sin la necesidad de estar conectado a un ordenador. Escanea todas las bandas de frecuencia WiMAX, tanto las licenciadas como las no licenciadas, mostrando instantáneamente si hay cobertura WiMAX. A través de la interfaz gráfica de usuario despliega las redes disponibles en la pantalla, así como indicadores del valor de la señal a ruido (SNR). Permite también al usuario posicionarse en el lugar más adecuado para conectarse. Se alimenta con una batería interna Li-On.

En la figura 5.5 mostramos tanto el dispositivo ‘MiMAX USB’ como el ‘MiMAX Finder’.

Hay otras marcas que también fabrican dispositivos de tipo USB para convertir un ordenador en WiMAX. Tenemos por ejemplo, el “USBw 100” de ‘Motorola’, que provee acceso inalámbrico en las bandas de 2,3GHz, 2,5 GHz y 3,5 GHz. Otro dispositivo muy similar sería el “Tu25” de ‘ZTE’.



Figura 5.5. Dispositivos “MiMAX” de ‘Airspan’.

### **5.2.2. ELECCIÓN DE LOS DISPOSITIVOS PARA EL LADO REMOTO.**

Una vez mencionadas los distintos dispositivos que nos podrían permitir conectar al profesor situado en remoto a Internet a través de WiMAX debemos escoger la mejor opción.

A priori escogeríamos un dispositivo móvil del tipo “SPH-9200” que ha fabricado ‘Samsung’, por varias razones. En primer lugar su reducido tamaño lo hace muy fácil y cómodo de transportar, y a la vez tiene un pantalla lo suficientemente grande como para que el profesor en remoto pueda ver lo que él mismo está transmitiendo hacia el otro extremo. Además posee una webcam, que lo ideal es que pudiese rotar para que el profesor pudiese enfocar lo que quisiera sin dejar de verse, pero este dispositivo concreto no nos da esa opción. De todos modos al ser un dispositivo que está dotado de varios puertos USB le podríamos conectar una cámara web externa apuntando a donde quisiésemos. Su autonomía de dos horas nos sería suficiente para llevar a cabo el tipo de videoconferencia que deseamos. Respecto al tema de la comunicación de audio bidireccional que se debe establecer entre ambos extremos, lo ideal sería conectar al ultra portátil un dispositivo que lleve integrado un auricular y un micrófono para asegurarnos de que el ruido ambiente no le impida al profesor en remoto escuchar las indicaciones que se le dan desde local, ni impida que los alumnos escuchen correctamente al profesor.

La situación estaría resuelta, pero desafortunadamente por el momento, no es posible disponer de este dispositivo en casi ningún lugar del mundo. Además, no se conoce su precio, con lo que no podemos asegurar que la solución planteada se ajuste cien por cien a los objetivos del proyecto; recordamos que uno de ellos es buscar una solución económica. Centrémonos entonces en conseguir una solución económica, efectiva y no demasiado aparatosa para el profesor en remoto, mientras esperamos que en unos años

dispositivos ultra móviles WiMAX como el mencionado hayan hecho aparición por nuestro continente.

La opción más viable y económica, será entonces utilizar un dispositivo de tipo *netbook* que nos permitirá ejecutar sobre él nuestro programa Teletrófono. Para permitir que este ordenador pueda acceder a Internet a través de WiMAX, le conectaremos uno de los dispositivos WiMAX USB de tipo ‘plug-and-play’ que se han mencionado. De este modo el profesor en remoto no necesitará llevar a cabo ninguna instalación ni complicada tarea, tan sólo enchufará el dispositivo USB en el *netbook* y listo. Todos los *netbooks* llevan integrada una cámara web, normalmente sobre la pantalla. Nosotros queremos que el profesor en remoto pueda estar viendo la pantalla de su ordenador mientras dirige la cámara a cualquier punto que desee. Para ello deberemos hacer lo que ya hemos comentado, conectar una cámara web por USB al *netbook* y situarla apuntando hacia donde queramos filmar. Del mismo modo, aunque los *netbooks* vienen equipados con micrófono y altavoces queremos optimizar la comunicación. Para evitar que el profesor en remoto no pueda oír bien las indicaciones que se le dan desde local debido al ruido del ambiente, o que este ruido dificulte que los alumnos en local oigan al profesor, utilizaremos otro dispositivo externo que consista en un auricular y un micrófono.

### 5.3. SOLUCIÓN FINAL.

Llegados a este punto del proyecto ya tenemos definida nuestra solución definitiva a la situación que planteábamos al inicio de este documento. Se planteó la situación concreta a resolver, se definieron unos objetivos y se fijaron unos requisitos, se estudió la tecnología que había que usar, se llevó a cabo el planteamiento y diseño del *software* de videoconferencia necesario y se implementó dicho programa. Por último hemos escogido los dispositivos que se usarán, sin olvidar nunca los objetivos iniciales.

En los anteriores apartados de este capítulo se han numerado los diferentes dispositivos que hemos considerado adecuado utilizar en cada uno de los lados de la comunicación, así que en este apartado vamos a mostrar simplemente una representación gráfica de la solución final.

En la siguiente figura, la 5.6, aparecen los dispositivos que hemos decidido usar en el aula donde se encuentra el profesor con los alumnos: un ordenador cualquiera para ejecutar en él nuestro programa Teletrófono (deberá tener acceso a Internet), altavoces para que los alumnos puedan oír bien la voz del profesor en remoto, un micrófono para poder hablar con dicho profesor y un proyector con su pantalla para que los alumnos puedan ver la pantalla de videoconferencia en la que aparecerá la imagen del profesor en remoto.



Figura 5.6. Dispositivos a utilizar en el lado local de la comunicación.

En esta otra figura, la 5.7, aparecen los dispositivos que deberá llevar consigo el profesor que se desplace a un entorno remoto: un auricular con micrófono integrado para comunicarse verbalmente con el otro extremo, un *netbook* de reducidas dimensiones que resulte cómodo de portar, una webcam que dirigiremos hacia lo que queramos filmar y un dispositivo USB WiMAX, que nos permitirá acceder a Internet a través de esta tecnología.



Figura 5.7. Dispositivos a utilizar en el lado remoto de la comunicación.



Por último, mostramos cómo podrían quedar los escenarios remoto y local con los dispositivos escogidos.



Figura 5.8. Representación de la solución final para el extremo remoto de la comunicación.



Figura 5.9. Representación de la solución final para el extremo local de la comunicación.





## **CAPÍTULO VI**

### **PRUEBAS REALIZADAS**

**\*Nota:** Tal y como se indicó al inicio del capítulo 3 de este proyecto, en un principio se pensaron realizar las pruebas con Teletrófono usando un acceso WiMAX, pero finalmente no fue posible. Por tanto, las pruebas se realizaron utilizando un acceso a la Red a través de Wi-Fi.

## 6.1. CLASE DE BOTÁNICA

Tras la implementación de Teletrófono se han realizado multitud de pruebas con éste, todas en entornos cerrados. Sin embargo se quiso realizar una prueba en concreto para demostrar realmente la utilidad de Teletrófono en un escenario como el que planteamos inicialmente, en el que cabía la posibilidad de que el extremo remoto se encontrase en un lugar al aire libre.

Las pruebas consistieron en comunicar a través de una videoconferencia Teletrófono a dos personas alejadas entre sí, recreando un escenario de videoconferencia educativa. El extremo local se situó en una habitación de una vivienda, en la que uno de los participantes en las pruebas se encargaría no sólo de participar en la videoconferencia, sino también de activar la herramienta de grabación de la sesión y de ir introduciendo subtítulos durante el transcurso de ésta. Hemos de imaginar que este participante es un profesor en un aula y que podría haber estado acompañado por alumnos. El otro extremo de la comunicación se situó en un parque, desde donde el otro participante en las pruebas simularía ser una profesora de ciencias naturales que pretendía impartir a los alumnos en el aula una clase de botánica a distancia. Ambos extremos estaban conectados a la misma red de área local.

Pasemos entonces a describir los materiales que se usaron para realizar esta prueba:

- En el lado local se utilizó un ordenador portátil, pero como si fuese un ordenador de sobremesa. Se enchufó a la red eléctrica y se ha conectó a través de cable a una red ADSL. El ordenador portátil que se usó es un “Acer Aspire 7720” de 17 pulgadas con micrófono y altavoces integrados.
- En el lado remoto se utilizó un *netbook* “Advent 4211” de 10 pulgadas con cámara web integrada y WiFi integrado. A éste le conectamos unos auriculares con micrófono de la marca “Plantronics”.

Nuestra aplicación Teletrófono lograba que:

- el participante situado en el parque transmitiese tanto sus palabras como la imagen que capturaba su webcam al participante situado en la vivienda,
- el participante situado en la vivienda pudiese también comunicarse verbalmente con su compañero situado en el parque,
- el participante en el parque pudiese ver en la pantalla de su propio *netbook* la imagen que estaba capturando en todo momento su webcam y, por tanto, la imagen que realmente estaba enviando a su compañero en local,

Además el participante situado en la vivienda en cuanto se inició la comunicación con el otro extremo, pulsó el botón de comenzar grabación. Este mismo extremo, durante la videoconferencia fue insertando subtítulos bajo la imagen que recibía del parque. Por

tanto, quedó grabada en su propia máquina la totalidad de la sesión de videoconferencia: en un archivo las imágenes que éste recibía del parque (archivo ‘.avi’), en otro archivo quedó grabado el audio que recibía de su compañero (archivo ‘.gsm’) y en otro archivo quedaron grabados los subtítulos que él mismo había ido insertando (archivo ‘.srt’).

Una vez finalizada la videoconferencia el participante del extremo local pudo reproducir, mediante el Reproductor Teletrófono, los archivos de audio y vídeo grabados así como los subtítulos, que se van mostrando en los mismos instantes de la sesión de videoconferencia que fueron introducidos. Tras la visualización, el participante en local acudió al Editor de Subtítulos Teletrófono para corregir alguna errata en estos.

En caso de que realmente el lado local de la videoconferencia hubiese estado situado en un aula con alumnos hubiésemos necesitado un proyector. Con el proyector hubiésemos podido mostrar a los alumnos la pantalla del ordenador, situado en este lado, en grandes dimensiones, de modo que hubiesen podido ver en grandes dimensiones, durante la videoconferencia, las imágenes del parque y de su profesora.

Con la prueba realizada pudimos comprobar que la aplicación funcionaba correctamente y que efectivamente, tal y como imaginábamos, podía ser de gran utilidad en entornos educativos. Además, gracias a las pruebas pudimos apreciar que:

- La sincronización del audio y el vídeo durante la videoconferencia no era perfecta, tal y como ya se nos advertía desde ‘Sun Microsystems’ (ver apartado 4.3 de este proyecto): “*Debido a las limitaciones de JMF 2.2.1, el audio y vídeo no estarán estrictamente sincronizados.*”. Esto afecta a la grabación de la sesión de videoconferencia, que grabará ambos flujos levemente desincronizados.
- El desfase entre el audio y vídeo recibidos puede aumentar en caso de bajas prestaciones (en concreto, memoria RAM muy limitada) del ordenador situado en el extremo remoto. Se ha comprobado que utilizando el *netbook* como extremo remoto se desfasa en mayor modo el audio y vídeo que si usamos un ordenador portátil con mayores prestaciones, lo que afecta también a la sincronización del audio y vídeo en la grabación de la sesión.
- La imagen y audio que se recibían en el lado local desde el parque tenían bastante calidad, a pesar de estar utilizando en el extremo remoto una cámara web que viene integrada en el *netbook*. Sin embargo, hemos de recordar, que a la hora de almacenar la grabación escogimos aquellos formatos que, aparte de permitirnos reproducir el audio y vídeo correctamente y lo más sincronizadamente posible, ocuparan el menor espacio en disco posible. Esto ha conllevado que la grabación pierda muchísima calidad con respecto a la imagen y audio que realmente se reciben en tiempo real durante la videoconferencia. Además como el códec de vídeo escogido es H.263, las dimensiones del vídeo se reducen a 352x288 píxeles, cuando durante la videoconferencia podemos normalmente disfrutar de unas dimensiones de imagen de 640x480 píxeles. Vemos que se reducen las dimensiones casi a la mitad.

En la figura 6.1 podemos observar cómo muestra el Reproductor Teletrófono el vídeo que la herramienta de grabación Teletrófono grabó durante la sesión de videoconferencia. Con el fotograma mostrado podemos ver que la imagen se muestra bastante pixelada. Se debe principalmente a la baja calidad de la webcam utilizada y a la codificación H.263 del vídeo en la grabación.



Figura 6.1. Reproductor Teletrófono reproduciendo el vídeo de las pruebas grabado (con la herramienta de grabación Teletrófono) de la sesión de videoconferencia.

Las pruebas realizadas que acabamos de mencionar fueron grabadas además con un par de cámaras de vídeo, para así poder dar constancia de ellas y que se pueda ver más claramente en qué consistieron. El montaje definitivo de este vídeo se mostrará durante la presentación y defensa de este proyecto.

## 6.2. VÍDEO DE EJEMPLO

Junto con el programa Teletrófono se proporciona un vídeo que también fue grabado durante una sesión de videoconferencia Teletrófono.

En esta ocasión las pruebas se realizaron entre dos habitaciones de una misma vivienda, porque lo que nos interesaba realmente no era comprobar la utilidad del programa en un entorno como el que este proyecto plantea, sino comprobar la calidad de grabación que se puede alcanzar usando en el lado remoto un ordenador con mayores prestaciones y que posee una webcam que proporciona mayor calidad de imagen que la webcam del *netbook*. Ambos extremos estaban, de nuevo, conectados a la misma red de área local.

En esta sesión de videoconferencia el participante en el lado remoto nos contó las utilidades de Teletrófono mientras el participante en el lado local (en otra habitación) fue insertando subtítulos. Esta prueba nos sirvió para comprobar la dependencia directa entre el retardo entre las pistas, y las prestaciones del hardware usado; también nos sirve para que los recién estrenados usuarios de Teletrófono que no hayan realizado todavía ninguna videoconferencia mediante esta aplicación puedan probar fácilmente el funcionamiento del Reproductor y el Editor de Subtítulos Teletrófono, utilizando como vídeo a reproducir, o a editar, el vídeo resultante de la grabación de esta sesión de videoconferencia.

Los materiales escogidos para llevar a cabo esta videoconferencia fueron:

- En el lado remoto se utilizó el ordenador “Acer Aspire” que habíamos usado en la anterior prueba como extremo local, que tiene webcam integrada, y al que le añadimos los auriculares y el micrófono “Plantronics” que también usamos en la anterior prueba.
- En el lado local se utilizó un Toshiba Satellite Pro L-300 de 15.4 pulgadas con micrófono y altavoces integrados.

Se pudo comprobar que el retardo introducido por el ordenador que situamos en el extremo remoto a la hora de enviar el audio y vídeo es prácticamente nulo, con lo que mejora notablemente la calidad de la videoconferencia y, por tanto, también de la grabación de ésta. A mayores prestaciones del ordenador usado en remoto, mayor calidad de la sesión de videoconferencia y, por tanto, de la grabación de ésta en cuanto a calidad de imagen y a minimización del retardo entre el audio y vídeo se refiere.

En la figura 6.2 que se muestra a continuación podemos ver cómo muestra el Reproductor Teletrófono el vídeo que la herramienta de grabación Teletrófono grabó durante la sesión de videoconferencia. Con el fotograma mostrado podemos ver que la imagen que se recibía desde remoto era de bastante calidad. Le mejora de la calidad de esta imagen con respecto a la que podíamos ver en la figura 6.1 se debe principalmente a la mejor calidad de la webcam utilizada.



Figura 6.2. Reproductor Teletrófono reproduciendo el vídeo de ejemplo grabado (con la herramienta de grabación Teletrófono) de la sesión de videoconferencia.

## **CAPÍTULO VII**

### **CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO**



## **7.1. CONCLUSIONES.**

Los principales objetivos que se plantearon al inicio de este documento han quedado resueltos. Hemos diseñado e implementado un programa de videoconferencia especialmente pensado para ser usado en centros educativos, hemos analizado la tecnología (WiMAX) que conecta ambos extremos de la videoconferencia, hemos desarrollado la aplicación a través del paquete JMF de Java, y hemos podido evaluar la utilidad de este paquete.

Se ha creado un programa de videoconferencia que puede resultar muy interesante a los docentes por su sencillez, facilidad de uso y posibilidades que ofrece. Esto podría ayudar a lograr que la videoconferencia se utilice como un método más de enseñanza, introduciéndola habitualmente en las aulas. Nuestro programa también puede resultar muy interesante a los alumnos, pues la introducción de rótulos puede asemejar la experiencia a un programa de televisión en directo, lo cual puede hacer que la experiencia les resulte más atractiva. Además, la grabación de las sesiones puede ofrecer a estos alumnos multitud de actividades, aparte de la propia participación en la videoconferencia.

Con este proyecto hemos conseguido asimismo, romper con el esquema de videoconferencia que se usa tradicionalmente en la educación, en el que los participantes están normalmente en habitaciones equipadas especialmente para albergar este tipo de actividades. Esto se ha logrado gracias principalmente a WiMAX. Se ha descubierto WiMAX como una tecnología especialmente útil para realizar este tipo de actividades. Este modo de acceso a la banda ancha inalámbrica, otorga libertad al usuario para poder estar en cualquier lugar, exterior o interior, y acceder a una conexión a Internet de alta velocidad (a mucha más velocidad de la que podemos disfrutar actualmente con las tecnologías de la tercera generación de móviles) a través de una antena WiMAX que puede estar situada a varios kilómetros de distancia. Más allá de si será WiMAX o no aquella tecnología que finalmente se impondrá en la guerra abierta entre las existentes y venideras tecnologías de banda ancha inalámbricas, resulta muy interesante saber que “hay vida” más allá de Wi-Fi (y sus múltiples versiones mejoradas) y de la finalmente exitosa 3G (y sus múltiples tecnologías añadidas a posteriori).

En definitiva, las funcionalidades de Teletrófono unidas a WiMAX permitirían introducir la videoconferencia en las aulas, multiplicando cuantiosamente las posibilidades y beneficios que dicha tecnología podría ofrecer tanto a educadores como a educandos. De hecho, esta unión, como hemos mencionado, provocaría que los propios alumnos situados en el aula pudiesen sentirse incluso como si estuviesen visionando un programa televisivo en directo con el que, además, pueden interaccionar. No podemos perder de vista entonces una aplicación muy importante que podría surgir de la unión de la videoconferencia y WiMAX: la sustitución de las unidades móviles de televisión y posibilidad de prescindir de los enlaces por satélite. A cambio se usaría la videoconferencia entre los estudios de televisión y los corresponsales, accediendo a la red a través de WiMAX. Como bien se ha comentado en el capítulo dos de este documento, algún programa de televisión ya se ha atrevido a prescindir de las costosas unidades móviles y enlaces por satélite en pro de la videoconferencia para conectar con sus corresponsales. Pero tal y como se puede apreciar en dichas conexiones, aún queda mucho por hacer. Las tecnologías y dispositivos a los que la población puede acceder en

la actualidad no son suficientes ni adecuados, para llevar a cabo este tipo de tareas. Sin embargo, el afamado y vertiginoso ritmo de cambio en las tecnologías de la información nos brindará, esperemos que pronto, la posibilidad de utilizar WiMAX, LTE y alguno más. Asimismo, en un futuro cercano, se diseñarán dispositivos preparados en exclusiva para comunicarse vía videoconferencia desde un remoto lugar cómodamente, ya sea para que el periodista transmita hasta el plató de televisión o el profesor hasta el aula donde tus alumnos ejercen de telespectadores.

Por otro lado sabemos que otro de los propósitos del proyecto era utilizar JMF para implementar la aplicación de videoconferencia, y poder evaluar así la utilidad y facilidad de uso este paquete de Java. Resumiendo mucho las cosas, podríamos decir que construir nuestra aplicación a través de JMF no nos ha resultado una tarea nada sencilla. A pesar de seguir escrupulosamente la API de dicho paquete, nos han surgido continuamente problemas, muchos de los cuales ya habían sido reportados por otros programadores a ‘Sun Microsystems’.

Tan sólo debemos “dar un paseo” por los numerosos foros y páginas web que hay en Internet sobre JMF para comprobar la impopularidad de éste. Y es que este paquete lleva abandonado por parte de sus creadores muchos años. En casi diez años, nadie ha actualizado los *codecs* y formatos con los que puede trabajar, nadie se ha preocupado en este tiempo en resolver los miles de *bugs* (errores) que se acumulan en su página web y, es más, sus propios creadores, listaron un total de veintitrés problemas conocidos (del inglés *known issues*) que tiene JMF, y que parece no van a ser solucionados nunca. Además en varios ejemplos de aplicaciones JMF que ‘Sun Microsystems’ tiene publicados en su página, ellos mismos mencionan importantes limitaciones de JMF. En definitiva nadie parece prestarle demasiada atención ni a estos problemas y graves limitaciones ni a la urgente actualización y mantenimiento que esta librería necesita.

Esta situación no nos ha impedido lograr nuestros objetivos a la hora de implementar nuestro programa pero sí nos ha creado problemas con los que hemos tenido que pelear para conseguir la aplicación que queríamos: Teletrófono.

## **7.2. FUTURAS LÍNEAS DE TRABAJO.**

Una vez finalizado el proyecto, es hora de plantearse qué mejoras podrían realizarse sobre éste en un futuro.

Decíamos en anteriores apartados que la herramienta de inserción de subtítulos de nuestro programa Teletrófono podía dar lugar a tantas actividades educativas como imaginación tuviese el profesor. Pues con las futuras líneas de trabajo a seguir ocurre lo mismo, hay tantas posibilidades como interesados pueda haber en alguno de los temas que este documento trata.

Con este proyecto, entre otras cosas, hemos creado un *software* de videoconferencia al que se le podrían añadir infinidad de herramientas a la imaginación del diseñador. De momento existe una herramienta de inserción de subtítulos, una de grabación de la sesión (y de los subtítulos que durante ella se insertan) y un editor de subtítulos. Se podrían añadir multitud de herramientas adicionales: una para enviar la grabación de la sesión por e-mail, otra para colgar la grabación en una página web, un editor de contenido del vídeo, etc.

Sin embargo existe una funcionalidad especialmente interesante que se podría añadir en futuras ocasiones a la aplicación Teletrófono, y es la posibilidad de conectar dos extremos situados en redes diferentes. De este modo se ampliarían las posibilidades de ubicación del extremo remoto. Para dotar a nuestra aplicación de esta posibilidad podríamos revisar las propuestas que se han explicado en el apartado 4.3 (“Problemas y conclusiones”) de este proyecto. Allí se cuenta el por qué de la imposibilidad a priori de comunicar mediante Teletrófono dos ordenadores que se encuentren en redes distintas, y además se indica cómo se podría dar solución a este problema en futuras revisiones de Teletrófono.

Otra de las funcionalidades que se podría añadir a Teletrófono sería la de implementar algún tipo de protocolo de señalización que permitiese prescindir de la utilización directa por parte de los usuarios de las direcciones IP de las máquinas. Lo deseable sería que los usuarios usaran, en vez de esas direcciones IP, direcciones lógicas que contuviesen sus respectivos nombres de usuario.

Más allá de nuestro programa de videoconferencia, y mencionando una vez más el acelerado ritmo de actualización tecnológica, podemos predecir que en breve periodo de tiempo surgirán varias posibilidades de actualización de la solución planteada por el proyecto. En primer lugar se deberá ver si WiMAX se impone finalmente como tecnología de masas en el mercado y, en un futuro, cuando la guerra entre WiMAX, LTE y quien sabe si algún actor de última hora más (seguro que aparecen muchos) haya finalizado, quizá se puedan ir perfilando esas nuevas soluciones para el acceso inalámbrico de banda ancha en el lado remoto de nuestra videoconferencia. Lo mismo ocurre con el hardware a utilizar en el proyecto. Probablemente en pocos años hayan aparecido dispositivos con mayores prestaciones, más ligeros e incluso pensados para escenarios como el que en este proyecto se plantea, o diseñados exclusivamente para la retransmisión en directo a un estudio de televisión, que podrían mejorar notablemente la solución planteada por este proyecto.

# Bibliografía

---

- 3G AMERICAS. The evolution of UMTS. 3GPP Release 5 and Beyond [en línea], Junio 2004, [ref. de 30 noviembre 2009].  
<[http://www.umtsforum.net/pdf/umtsrel5\\_beyond\\_june2004.pdf](http://www.umtsforum.net/pdf/umtsrel5_beyond_june2004.pdf)>
- ACTUALIDAD INFORMÁTICA. *Tres tecnologías se disputan el futuro de los celulares* [en línea], Febrero 2008, [ref. de 29 noviembre 2009].  
<<http://www.marisolcollazos.es/noticias-informatica/index.php?s=huawei>>
- ADA-MADRID. [en línea], [ref. de 15 septiembre 2009].  
<<http://moodle.upm.es/adamadrid/>>
- ADSL AYUDA. *Introducción al NAT* [en línea], [ref de 10 febrero 2010].  
<<http://www.adslayuda.com/Generico-nat.html>>
- AMIMO RAYOLLA et al. *Wireless Broadband: Comparative Analysis of HSDPA vs. WiMAX* [en línea], [ref. de 15 noviembre 2009].  
<<http://home.intekom.com/satnac/proceedings/2007/papers/access/Paper%2078%20-%20Amimo-Rayolla.pdf>>
- ARAMBERRI, J. et al. *Teleformación síncrona. Experiencias con sistemas de videoconferencia en ATM* [en línea], [ref. de 26 junio 2009].  
<<http://www.rediris.es/difusion/publicaciones/boletin/50-51/ponencia14.html>>
- ASOCIACIÓN ESPAÑOLA DE COMUNICACIONES MÓVILES. *Adiós a las unidades móviles de Televisión* [en línea], Julio 2007, [ref. de 2 julio 2009].  
<<http://www.aecomo.org/content.asp?ContentTypeID=2&ContentId=8078&CatTypeID=2&CatID=269>>
- BBC NEWS. *Intel seeks wireless unification* [en línea], Junio 2008, [ref. de 3 enero 2010]. <<http://news.bbc.co.uk/2/hi/technology/7425756.stm>>
- BELLIDO, Luis. *A survey on standards for videoconferencing over broadband networks* [en línea], [ref. de 20 octubre 2009].  
<<http://greco.dit.upm.es/~leverage/conf1/bellido.htm>>
- BROADCOM. *802.11n: Next-Generation Wireless LAN Technology* [en línea], Abril 2006, [ref. de 19 octubre 2009]. <[http://80211n.com/white\\_paper/802\\_11n-WP100-R.pdf](http://80211n.com/white_paper/802_11n-WP100-R.pdf)>
- CHACÓN MEDINA, Antonio. *La videoconferencia: conceptualización, elementos y uso educativo* [en línea], [ref. de 17 julio 2009]. Etic@ net, Año 1, Número 2, Diciembre de 2003, ISSN: 1695-324X. <  
<http://www.teleformacion.edu/documentos/vc.pdf><http://www.ugr.es/>>

- CHASCO, LÓPEZ y GONZÁLEZ. *El e-learning en la universidad española* [en línea], 2003, [ref. de 29 mayo 2009].  
<<http://www.asepelt.org/ficheros/File/Anales/2003%20-%20Almeria/asepeltPDF/112.PDF>>
- CLEARWIRE. [en línea] [ref. de 6 julio 2009]. <<http://www.clearwire.es/>>
- CLUB DE INNOVACIÓN. *El Ayuntamiento de Alcorcón invierte en tecnología WiMax y vídeo IP para mejorar la seguridad ciudadana y de sus edificios públicos* [en línea], [ref. de 10 agosto 2009].  
<[http://www.clubdeinnovacion.es/index.php?option=com\\_mtree&task=viewlink&link\\_id=76&Itemid=80](http://www.clubdeinnovacion.es/index.php?option=com_mtree&task=viewlink&link_id=76&Itemid=80)>
- CONFERENCE XP, [programa informático] <<http://research.microsoft.com/en-us/projects/conferencexp/>>
- CORRAL Carlos; CRUZ, Francisco. *Videoconferencia y tele-educación en la Universidad Carlos III de Madrid: Infraestructuras* [en línea], [ref. de 12 junio 2009].  
<<http://www.rediris.es/difusion/publicaciones/boletin/50-51/ponencia15.html>>
- DESARROLLO WEB. *La CMT autoriza bajo ciertas condiciones que la EMT ofrezca Internet en su red de autobuses públicos* [en línea], Marzo 2009, [ref. de 11 julio 2009]. <<http://www.desarrolloweb.com/actualidad/wifi-publica-autobuses-madrid-1521.html>>
- DIMDIM, (programa informático) <<http://www.dimdim.com/>>
- DOMÍNGUEZ, Luis. *¿Qué es WiMAX?* [en línea], Diciembre 2008, [ref. de 1 febrero 2010]. <[http://articulos-blackhat4all.blogspot.com/2008/12/que-es-wimax.html?utm\\_source=feedburner&utm\\_medium=feed&utm\\_campaign=Feed%3A+BlackHat-Articulos+\(Black+Hat+-+Art%C3%ADculos\)](http://articulos-blackhat4all.blogspot.com/2008/12/que-es-wimax.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+BlackHat-Articulos+(Black+Hat+-+Art%C3%ADculos))>
- DOMÍNGUEZ MASCARELL, Amparo. *Estándares de Videoconferencia* [en línea], [ref. de 30 junio 2009].  
<<http://www.uv.es/montanan/redes/trabajos/videoconferencia.doc>>
- EASYNET. *Face to Phase: La convergencia entre los viajes de negocio y las reuniones virtuales* [en línea], Enero 2009, [ref. de 3 septiembre 2009].  
<<http://www.easynet.com/es/es/about/pressRelease.aspx?TertiaryNavID=777&PressReleaseID=929>>
- EKIGA, (programa informático) <<http://www.ekiga.org/>>
- EL PAÍS. *Los jueces piden la videoconferencia para que declaren peritos y testigos* [en línea], 19 Mayo 2001, [ref. de 30 mayo 2009].  
<[http://www.elpais.com/articulo/madrid/jueces/piden/videoconferencia/declaren/peritos/testigos/elpepiespmad/20010519elpmad\\_13/Tes](http://www.elpais.com/articulo/madrid/jueces/piden/videoconferencia/declaren/peritos/testigos/elpepiespmad/20010519elpmad_13/Tes)>
- ESPAÑA. MINISTERIO DE ASUNTOS EXTERIORES Y COOPERACIÓN. *PCI Intercampus* [en línea], [ref. de 5 septiembre 2009].  
<<http://www.becasmae.es/pci/>>
- ESPAÑA. MINISTERIO DE EDUCACIÓN. *Administrar la red en un IES* [en línea], Abril 2009, [ref. de 10 noviembre 2009].

<<http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=694>>

- EUSKALTEL. [en línea] [ref. de 3 agosto 2009]. <<http://euskaltel.com/>>

- EXTREMADURA AL DÍA. *El IES Hernández Pacheco de Cáceres se conectará por videoconferencia con el Hospital San Pedro de Alcántara* [en línea], 12 Diciembre 2006, [ref. de 13 julio 2009]. <<http://www.extremaduraaldia.com/tecnologia/el-ies-hernandez-pacheco-de-caceres-se-conectaran-por-videoconferencia-con-el-hospital-san-pedro-de-alcantara/29629.html>>

- FELICI, Santiago. *Videoconferencia* [en línea], [ref. de 13 julio 2009]. <<http://informatica.uv.es/doctorado/SST/docto-4-voip.ppt>>

- FLASH MEETING, (programa informático) <<http://flashmeeting.open.ac.uk/>>

- GAPTEL. *Comunicaciones móviles e inalámbricas* [en línea], Septiembre 2005, [ref. de 22 agosto 2009]. <<http://www.observatorioaragones.org/estudios/gaptelmoviles.pdf>>

- GARCÍA FERNÁNDEZ, Marinela. *La videoconferencia aplicada a la enseñanza del español técnico en proyectos internacionales* [en línea], [ref. de 6 junio 2009]. <[http://cvc.cervantes.es/ensenanza/biblioteca\\_ele/ciefe/pdf/01/cvc\\_ciefe\\_01\\_0013.pdf](http://cvc.cervantes.es/ensenanza/biblioteca_ele/ciefe/pdf/01/cvc_ciefe_01_0013.pdf)>

- GEARFUSE. *US-based WiMAX network ready to launch* [en línea], 2008, [ref. de 28 septiembre 2009]. <<http://www.gearfuse.com/us-based-wimax-network-ready-to-launch/>>

- GNANACHANDRAN, Janahan; PRASHANTH REDDY BONTHU, Venkata. *Mobile Broadband Wireless Access – IEEE 802.16 vs 802.20* [en línea], [ref. de 18 octubre 2009]. <<http://www.docstoc.com/docs/14924328/MOBILE-BROADBAND-WIRELESS-ACCESS-%E2%80%93-IEEE-80216-vs-80220>>

- GRUPO 9 UNIVERSIDADES. *Preguntas más frecuentes*, [en línea], [ref. de 30 julio 2009]. <<https://www.uni-g9.net/portal/faq.html>>

- HERNÁNDEZ ABADÍA DE BARBARÁ, Alberto. *Sistema de Telemedicina de la Fuerzas Armadas españolas* [en línea], Mayo 2006, [ref. de 29 julio 2009]. <[http://www.csi.map.es/csi/tecniemap/tecniemap\\_2006/05T\\_PDF/sistema%20de%20telemedicina.pdf](http://www.csi.map.es/csi/tecniemap/tecniemap_2006/05T_PDF/sistema%20de%20telemedicina.pdf)>

- HUIDOBRO, José Manuel. *WiMAX. ¿El sustituto de Wi-Fi?* [en línea], [ref. de 16 octubre 2009]. <<http://www.monografias.com/trabajos16/wimax/wimax.shtml>>

- H.323 FORUM, *H.323 Standards* [en línea], [ref. de 28 mayo 2009]. <<http://www.h323forum.org/>>

- IBERBANDA. [en línea] [ref. de 7 septiembre 2009]. <<http://www.iberbanda.es/>>

- IBERSYSTEMS. *Proyecto: Red WiMAX de cobertura extensa para control de regadío en Almería* [en línea], [ref. de 22 julio 2009]. <[http://www.ibersystems.es/cobertura\\_red\\_wimax\\_nijar\\_control\\_de\\_regadio.aspx#](http://www.ibersystems.es/cobertura_red_wimax_nijar_control_de_regadio.aspx#)>

- IBERSYSTEMS. *Redes inalámbricas- Tecnología WiMAX* [en línea], [ref. de 22 julio 2009]. <[http://www.ibersystems.es/tecnologia\\_wimax.aspx](http://www.ibersystems.es/tecnologia_wimax.aspx)>
- IBERSYSTEMS. *Unión WiMAX redundante de alta velocidad (140 Mbps)* [en línea], [ref. de 22 julio 2009]. <[http://www.ibersystems.es/ultimos\\_proyectos\\_wimax\\_union.aspx](http://www.ibersystems.es/ultimos_proyectos_wimax_union.aspx)>
- IBM. *IBM Certified Course: Tecnología WiMAX, 2006* (Código del curso: RR830ES).
- IDG COMMUNICATIONS. *LMDS en marcha* [en línea], [ref. de 2 diciembre 2009]. <<http://www.idg.es/comunicaciones/impart.asp?id=117311>>
- IEEE. *Introduction to IEEE 802.20* [en línea], Marzo 2003, [ref. de 30 noviembre 2009]. <[http://www.ieee802.org/20/P\\_Docs/IEEE%20802.20%20PD-04.pdf](http://www.ieee802.org/20/P_Docs/IEEE%20802.20%20PD-04.pdf)>
- IEEE Standards Association. *IEEE 802.16 Broadband Wireless Metropolitan Area Network* [en línea], [ref. de 30 noviembre 2009]. <<http://standards.ieee.org/getieee802/802.16.html>>
- INFOMED. RED TELEMÁTICA DE SALUD EN CUBA. *Educación a Distancia: ¿Para qué y cómo?* [en línea], [ref. de 3 mayo 2009]. <<http://www.sld.cu/libros/distancia/cap3.html#Perfil>>
- INFORMADOR. *Nokia deja de producir su único móvil con WiMAX* [en línea], Enero 2008, [ref. de 11 septiembre 2009]. <<http://www.informador.com.mx/tecnologia/2009/68644/6/nokia-deja-de-producir-su-unico-movil-con-wimax.htm>>
- INSTITUTO UNIVERSITARIO DE POSGRADO. *Quiénes somos* [en línea], [ref. de 3 junio 2009]. <<http://www.iup.es/esl/sobre-iup>>
- INTERNATIONAL TELECOMMUNICATION UNION. *ITU-T Recommendations: H series* [en línea], [ref. de 1 octubre 2009]. <<http://www.itu.int/>>
- iXBT LABS. *HSDPA vs. WiMAX: Comparing Characteristics and Prospects of Datacom Technologies* [en línea], Diciembre 2006, [ref. de 4 noviembre 2009]. <<http://ixbtlabs.com/articles2/mobile/wimax.html>>
- JAVA WORLD. *Solutions for Java developers* [en línea], [ref. de 8 Diciembre 2009]. <<http://www.javaworld.com>>
- JAVARANCH FORUMS. [en línea] [ref. de 28 noviembre 2009]. <<http://www.coderanch.com/>>
- KATZ, Marcos D; FITZEK, Frank H.P. *WiMAX evolution: emerging technologies and applications*, Chichester: John Wiley & Sons, 2009, 468 p., ISBN 9780470696804.
- KNOWLEDGE ON DEMAND PROYECT. *Project Description* [en línea], [ref. de 12 mayo 2009]. <<http://kod.iti.gr/>>
- KOLABORA. *Flash-based Video Conferencing Solutions On The Web* [en línea], [ref. de 3 mayo 2009].

<[http://www.kolabora.com/news/2004/11/18/flashbased\\_video\\_conferencing\\_solutions\\_on.htm](http://www.kolabora.com/news/2004/11/18/flashbased_video_conferencing_solutions_on.htm)>

- KPHONE, (programa informático) <<http://sourceforge.net/projects/kphone/>>

- LA VOZ DIGITAL. *Las cámaras toman el quirófano* [en línea], Abril 2009, [ref. de 15 junio 2009].

<[http://www.lavozdigital.es/cadiz/20090415/puerto\\_real/camaras-toman-quiroyfano-20090415.html](http://www.lavozdigital.es/cadiz/20090415/puerto_real/camaras-toman-quiroyfano-20090415.html)>

- LAMELAS Torrijos, Andrés. *WiMAX* [en línea], 2006, [ref. de 19 octubre 2009].

<<http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=349>>

- LINPHONE, (programa informático) <<http://www.linphone.org/>>

- LÓPEZ CEREZO, José A. *Ciencia y Tecnología en Sociedad* [en línea], Octubre 2009, [ref. de 26 septiembre 2009].

<<http://www.madridiario.es/jorgejuan/noticia/2009/octubre/opinion/jose-antonio-lopez-cerezo/109742/>>

- MÁRQUEZ, Emilio. *La videoconferencia, una opción ante la crisis* [en línea], Febrero 2009, [ref. de 30 septiembre 2009]. <<http://www.unblogenred.es/la-videoconferencia-una-opcion-ante-la-crisis/>>

- MATA, Manuel. *El estándar H. 323 para comunicación multimedia* [en línea], [ref. de 22 junio 2009]. <<http://neutron.ing.ucv.ve/revista-e/No5/MMata.htm>>

- MERCURO IMS CLIENT, (programa informático) <<http://www.mercurio.net/>>

- MIDLAND COMMUNICATIONS. *The Sophia Home* [en línea], [ref. de 26 septiembre 2009]. <<http://www.midlandcommsvc.co.uk/>>

- MINISIP, (programa informático) <<http://www.minisip.org/>>

- MORAL, José Antonio. *El Ayuntamiento de Gata instalará cámaras conectadas por wimax para prevenir el vandalismo* [en línea], Marzo 2009, [ref. de 14 junio 2009].

<<http://blogwimax.com/2009/03/01/el-ayuntamiento-de-gata-instalara-camaras-conectadas-por-wimax-para-prevenir-el-vandalismo/>>

- MORAL, José Antonio. *La Universidad de Cádiz instala una red wimax* [en línea], Agosto 2007, [ref. de 14 junio 2009]. <<http://blogwimax.com/2007/08/07/la-universidad-de-cadiz-instala-una-red-wimax/>>

- MORAL, José Antonio. *Valdepeñas pondrá en marcha una red WiMAX-WiFi* [en línea], Marzo 2009, [ref. de 14 junio 2009].

<<http://blogwimax.com/2009/03/22/valdepenas-pondra-en-marcha-una-red-wimax-wifi/>>

- MUÑOZ, Irazú. *WiMAX: El nuevo acceso inalámbrico a Internet* [en línea], 2005, [ref. de 22 agosto 2009]. <<http://www.cinit.org.mx/articulo.php?idArticulo=31>>

- MUÑOZ ORGANERO, Mario. *Players* [en línea]. Asignatura: *Servidores de Información Multimedia*, Departamento de ingeniería Telemática, Universidad



Carlos III de Madrid, [ref. de 5 febrero 2010].

<[http://www.it.uc3m.es/~labsimitis/sesiones/teoria/03\\_Players.pdf](http://www.it.uc3m.es/~labsimitis/sesiones/teoria/03_Players.pdf)>

- NASA, *Glenn Research Center* [en línea], [ref. de 4 julio 2009].

<<http://www.nasa.gov/centers/glenn/home/index.html>>

- NEURORRADIOLOGÍA. *Los hospitales de Alicante y Vega Baja realizan sesiones clínicas de patología cerebrovascular por videoconferencia* [en línea], Marzo 2008, [ref. de 14 junio 2009]. <<http://www.neurorradiologia.org/sesiones/>>

- NEO-SKY. [en línea] [ref. de 2 mayo 2009]. <<http://www.neo-sky.com/>>

- NOTICIAS CADA DÍA. *Los municipios de la mancomunidad La Mancha estarán intercomunicados a través de una infraestructura de banda ancha inalámbrica* [en línea], Marzo 2008, [ref. de 13 junio 2009].

<<http://www.noticiascadadia.com/noticia/11759-los-municipios-de-la-mancomunidad-la-mancha-estaran-intercomunicados-a-traves-de-una-infraes/>>

- NOTICIAS.INFO. *Barcelona Televisió y Telefónica I+D transmiten TV en directo sin necesidad de unidades móviles* [en línea], Julio 2007, [ref. de 30 julio 2009].

<[http://www.noticias.info/archivo/2007/200707/20070727/20070727\\_303739.shtm](http://www.noticias.info/archivo/2007/200707/20070727/20070727_303739.shtm)>

- OHRTMAN, Frank. *WiMAX handbook : building 802.16 wireless networks*, McGraw-Hill, 2005, 261 p., ISBN: 0071454012.

- OLIVARES, Javier (Diario de Mallorca, 08/05/01). *El primer juicio en España por videoconferencia se celebra desde la UIB con normalidad* [en línea], [ref. de 24 mayo 2009]. <<http://www.uib.es/premsa/maig01/dia-08/332519.htm>>

- OLIVER ,M. *La videoconferencia en el campo educativo. Técnicas y procedimientos* [en línea], [ref. de 29 junio 2009].

<<http://uib.es/depart/gte/oliver.html>>

- ONTARIO TELEMEDICINE NETWORK. *Who we are* [en línea], [ref. de 24 mayo 2009]. <<http://www.otn.ca/en/otn/who-we-are/>>

- OOVOO, (programa informático) <<http://www.oovoo.com/>>

- PABLO FERNÁNDEZ. *La videoconferencia matará a los viajes en avión* [en línea], Enero 2009, [ref. de 5 agosto 2009]. <<http://www.eweekurope.es/noticias/la-videoconferencia-matara-a-los-viajes-en-avion-486>>

- PALBEE, (programa informático) <<http://www.palbee.com/>>

- PAZMIÑO, F. *Videoconferencia* [en línea], [ref. de 3 diciembre 2009].

<<http://www.monografias.com/trabajos/videoconferencia/videoconferencia.shtml>>

- PIERCE, John R.; NOLL, A. Michael. *Señales. La Ciencia de las Telecomunicaciones*, Editorial Reverté, S.A., 1995, 255p., ISBN 8429143874

- PINEDA MARÍN, Fabián Albeiro. *WiMAX: Presente y futuro del acceso a banda ancha inalámbrica en Colombia* [en línea], [ref. de 11 diciembre 2009].

<<http://www.monografias.com/trabajos60/wimax-banda-ancha/wimax-banda-ancha.shtml>>

- PRADES DEL VALLE, Carlos. *Tratamiento Multimedia en Java con JMF* [en línea], versión 1.0.1, Febrero 2001, [ref. de 5 febrero 2010].  
<<http://cprades.eresmas.com/Tecnica/programarJMF.pdf>>
- QNEXT, (programa informático) <<http://www.qnext.com/>>
- QUOCIRCA. *Visual Impact- the emerging face of business collaboration* [en línea], Marzo 2007, [ref. de 3 julio 2009].  
<<http://www.quocirca.com/pages/analysis/reports/view/store250/item3702/>>
- RADIOPTICA. *Redes WiMAX* [en línea], [ref. de 4 octubre 2009].  
<<http://www.radioptica.com/wimax/>>
- REDES TELECOM. *La guardia urbana de Vic implanta una red de vigilancia con las soluciones de Alvarion* [en línea], Enero 2009, [ref. de 14 junio 2009].  
<<http://www.redestelecom.es/Noticias/200901270018/La-Guardia-Urbana-de-Vic-implanta-una-red-de-vigilancia-con-las-soluciones-de-Alvarion.aspx>>
- REVISTA INSUMOS & INSTRUMENTOS #9. *Telemedicina: Una Aplicación Multimedia* [en línea], Febrero 2001, [ref. de 20 mayo 2009].  
<[http://www.bioingenieros.com/empresas/historial/revista\\_9.txt](http://www.bioingenieros.com/empresas/historial/revista_9.txt)>
- RTVE. *Corresponsales en 24 horas* [en línea], [ref. de 20 abril 2009].  
<<http://blogs.rtve.es/corresponsales/posts>>
- SIGALÉS, Carles et al. *La integración de Internet en la educación escolar española: situación actual y perspectivas de futuro* [en línea], Julio 2008, [ref. de 1 mayo 2009].  
<[http://www.oei.es/salactsi/informe\\_escuelas\\_espana\\_09.pdf](http://www.oei.es/salactsi/informe_escuelas_espana_09.pdf)>
- SIGHT SPEED, (programa informático) <<http://www.sightspeed.com/>>
- SKYPE, (programa informático) <<http://www.skype.com/>>
- SONY. *La videoconferencia reduce el riesgo de infección en una clínica sueca* [en línea], Diciembre 2007, [ref. de 5 julio 2009].  
<[http://www.sony.es/biz/view/ShowContent.action?site=biz\\_es\\_ES&contentId=1196773901899&sectiontype=HC+CaseStudies](http://www.sony.es/biz/view/ShowContent.action?site=biz_es_ES&contentId=1196773901899&sectiontype=HC+CaseStudies)>
- SUN DEVELOPER NETWORK. *Code sample: Transcoding to Different Formats* [en línea], [ref. de 3 diciembre 2009].  
<<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/Transcode.html>>
- SUN DEVELOPER NETWORK. *Documentation: Known Issues* [en línea], [ref. de 22 diciembre 2009].  
<<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/issues.html>>
- SUN DEVELOPER NETWORK. *Frequently Asked Questions* [en línea], [ref. de 22 diciembre 2009].  
<<http://java.sun.com/javase/technologies/desktop/media/jmf/reference/faqs/index.html#jmf2-sync>>

- SUN DEVELOPER NETWORK. *JMF 2.1.1- Supported Formats* [en línea], [ref. de 7 enero 2010].  
<<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/formats.html>>
- SUN FORUMS. [en línea] [ref. de 28 noviembre 2009].  
<<http://forums.sun.com/>>
- SUN MICROSYSTEMS. *Discussion List for Java Media Framework API* [en línea], [ref. de 2 enero 2010]. <<http://archives.java.sun.com/archives/jmf-interest.html>>
- SUN MICROSYSTEMS. *Java Media Framework API Guide* [en línea], [ref. de 3 febrero 2010].  
<<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/>>
- TANDBERG. *Discussion Document: Assessing the Real Impact on Business Travel* [en línea], [ref. de 10 agosto 2009].  
<[http://www.tandberg.com/collateral/tandberg\\_videoconfering\\_travel\\_survey.pdf](http://www.tandberg.com/collateral/tandberg_videoconfering_travel_survey.pdf)>
- TELESEMANA. *WiMAX vs. HSDPA* [en línea], [ref. de 5 diciembre 2009].  
<<http://www.scribd.com/doc/11491172/WIMAX-vs-3G>>
- TIC PYMES. *El clímax de la videoconferencia* [en línea], Enero 2009, [ref. de 30 mayo 2009]. <<http://www.ticpymes.es/Noticias/General/200901300012/-El-climax-de-la-videoconferencia-.aspx>>
- TICPYMES. *Sagunto se convierte en el mejor municipio impulsor de las TIC* [en línea], Junio 2009, [ref. de 1 julio 2009].  
<<http://www.ticpymes.es/Noticias/General/200906080013/Sagunto-se-convierte-en-el-mejor-municipio-impulsor-de-las-TIC.aspx>>
- TIOBE. *TIOBE Programming Community Index for December 2009* [en línea], Diciembre 2009, [ref. de 15 diciembre 2009].  
<<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>>
- TOKBOX, (programa informático) <<http://www.tokbox.com/>>
- UNIVERSIA. *La Escuela Técnica Superior de Ingenieros de la Universidad de Sevilla desarrolla un prototipo para realizar directos de televisión sin la intermediación de un vehículo de unidad móvil* [en línea], Enero 2009, [ref. de 2 agosto 2009].  
<[http://www.universia.es/html\\_estatico/portada/actualidad/noticia\\_actualidad/param/noticia/jifdb.html](http://www.universia.es/html_estatico/portada/actualidad/noticia_actualidad/param/noticia/jifdb.html)>
- UNIVERSIDAD CARLOS III DE MADRID. ÁREA DE AUDIOVISUALES. *Videoconferencia y tele-educación en la Universidad Carlos III* [en línea], [ref. de 15 mayo 2009].  
<<http://audiovisuales.uc3m.es/modules.php?name=Content&pa=showpage&pid=8>>
- UNIVERSITAT DE BARCELONA. *Intercampus UB. Presentació* [en línea], [ref. de 9 noviembre 2009].  
<<http://www.ub.es/acad/lliure/assignatures/intercampus.htm>>

- UNIVERSITAT OBERTA DE CATALUNYA. [en línea], [ref. de 9 noviembre 2009].  
<<http://www.uoc.edu/web/esp/index.html>>
- VALDOSTA STATE UNIVERSITY. *American Sign Language Interpreting and Deaf Education* [en línea], [ref. de 17 julio 2009].  
<<http://www.valdosta.edu/coe/sec/ASLInterpretingandDeafEducation.shtml>>
- VARAS C., Sofía. *Videoconferencia al servicio de la Justicia* [en línea], [ref. de 24 mayo 2009]. <[http://www.latintele.com/fileadmin/pdfs\\_latintele/noticias](http://www.latintele.com/fileadmin/pdfs_latintele/noticias)>
- VILA. Pablo. *Pre-WiMAX o Post- WiFi: ¿Nos están engañando?* [en línea], Diciembre 2008, [ref. de 3 septiembre 2009].  
<<http://albentia.wordpress.com/2008/12/06/pre-wimax-o-post-wifi-%C2%BFnos-estan-enganando/>>
- VoIP FORO. *Protocolos VoIP* [en línea], [ref. de 10 agosto 2009].  
<<http://www.voipforo.com/protocolosvoip.php>>
- VoIP-LIST. *VoIP providers List* [en línea], [ref. de 10 agosto 2009].  
<<http://www.voip-list.com/>>
- VSEE, (programa informático) <<http://www.vsee.com/>>
- VYEW, (programa informático) <<http://www.vyew.com/>>
- WASON, Terry. *WiMAX Networks* [en línea], [ref. de 28 septiembre 2009].  
<<http://www.sanog.org/resources/sanog5-terry-wimax.pdf>>
- WiMAX FORUM, [en línea], [ref. de 30 noviembre 2009].  
<<http://www.wimaxforum.org/>>
- WINDOWS LIVE MESSENGER, (programa informático)  
<<http://download.live.com/>>
- YAHOO! MESSENGER, (programa informático)  
<<http://es.messenger.yahoo.com/>>



# ANEXO I

## MANUAL DE USUARIO DE TELETRÓFONO

### Introducción

¿Qué es Teletrófono?

Teletrófono es un software de videoconferencia pensado especialmente para su uso en un entorno educativo y que permite realizar tareas adicionales a la mera transmisión de audio y vídeo entre los extremos.

Con él, podremos conectar dos lugares alejados físicamente entre sí. Así, por ejemplo, desde un aula de un centro educativo, los alumnos podrán ver en directo, como si de un programa de televisión se tratara, a uno de sus profesores relatando datos de interés acerca del lugar al que se ha desplazado. La interactividad está garantizada, pues la comunicación de audio se realiza en los dos sentidos de la comunicación, para que así los alumnos puedan plantearle sus dudas al profesor en cualquier momento. Además el profesor que se encuentra en el aula con los alumnos podrá ir insertando rótulos bajo la imagen que reciben en directo. Teletrófono nos permite también grabar la videoconferencia con sus subtítulos, almacenar dicha grabación, reproducirla o incluso modificar los subtítulos que se insertaron. A través de este manual aprenderemos a hacer todo esto y mucho más.

Es importante saber que a lo largo del manual aparecerán en ocasiones los símbolos:



y

El primero indica que el texto que lo acompaña es una indicación muy importante para el correcto funcionamiento del programa; el segundo símbolo nos indica que el texto que le sigue proporciona información adicional para un mejor aprovechamiento de Teletrófono o una aclaración.

Ahora sí, comencemos.

## Instalación de Teletrófono

Antes de comenzar a usar Teletrófono debemos instalar el programa en nuestro ordenador. Vamos a ofrecer tres formas distintas para poder comenzar a usar Teletrófono:

**\* Nota 1:** Junto con las tres carpetas que nos permitirán escoger tres opciones distintas de ejecutar Teletrófono, se proporciona una carpeta llamada 'Vídeo-ejemplo', que contiene la grabación de una sesión Teletrófono. Con estos archivos podremos probar inmediatamente el Reproductor y el Editor de Subtítulos Teletrófono, aunque todavía no hayamos llevado a cabo ninguna videoconferencia con Teletrófono.

**OPCIÓN 1:** Si ya teníamos instalado Java y JMF (Java Media Framework) en nuestro ordenador, podremos ejecutar Teletrófono abriendo la carpeta 'OPCION1-jar' y haciendo doble clic en "Teletrófono.jar". De inmediato se lanzará nuestro programa.

\* A veces esta opción puede darnos problemas según el sistema operativo utilizado. El problema más común es que no detecte dispositivos de captura. En este caso se recomienda utilizar la opción de instalación número dos.

**OPCIÓN 2:** Si ya teníamos instalado Java y JMF en nuestro ordenador, podemos ejecutar Teletrófono disponiendo de una carpeta con todos los archivos necesarios para ello. Esta carpeta la hemos llamado 'OPCION2-class'. Para ejecutar Teletrófono de este modo deberemos acudir al Símbolo del Sistema (cmd) y, desde allí, situarnos en la citada carpeta, llamada 'OPCION2-class'. Después, también en el Símbolo del Sistema, teclearemos la sentencia: 'java Presentacion'. ('Presentacion' es la clase que contiene el 'main' de nuestro programa).

**OPCIÓN 3:** Si no tengo instalado Java o JMF en mi ordenador, tenemos dos opciones. La primera de ellas es instalar Java y JMF (ver '\*Nota 2') y la segunda es utilizar un instalador de Teletrófono. Para llevar a cabo la segunda opción, debemos:

**a)** Abrir la carpeta 'OPCION3-instalador' y hacer doble clic sobre 'Teletrófono\_windows\_1\_1.exe' (es un instalador sólo válido para Windows). Esta acción hará que, pasados unos segundos, se inicie el asistente de instalación de Teletrófono.

- La primera pantalla del asistente nos aconseja cerrar el resto de aplicaciones que haya ejecutándose en nuestro ordenador. Pulsamos entonces el botón "Siguiente".

- La segunda pantalla del asistente nos muestra el directorio donde se instalará por defecto Teletrófono. Si queremos cambiar dicho directorio podemos hacerlo manualmente o a través del botón "Explorar". Pulsamos el botón "Siguiente".

- La tercera pantalla del asistente nos indica que se creará una carpeta en el menú Inicio de nuestro ordenador que por defecto se llamará Teletrófono. Podemos cambiar el nombre de dicha carpeta o indicar al asistente que no queremos crear dicha carpeta. Cuando pulsemos ahora el botón "Siguiente" se iniciará la instalación de Teletrófono, apareciendo una pantalla que muestra la evolución del proceso de instalación.

- Por último aparece una pantalla que nos indica que la instalación ha finalizado con éxito y que podemos ejecutar la aplicación a través de los accesos directos creados. Ya podemos pulsar el botón “Finalizar”, que cerrará el asistente de instalación.

**b)** Acudo a las Variables de Entorno de mi Sistema e indico en la variable ‘PATH’ la ruta donde se encuentra Java. Por ejemplo, si hemos dejado la ruta por defecto que el instalador de Teletrófono nos ofrecía, en la variable ‘PATH’ deberemos introducir: “C:\Archivos de programa\Teletrófono\jre\bin”.

**c)** Ahora debo acudir a una de las carpetas situadas dentro de mi directorio de instalación Teletrófono. Es la carpeta ‘bin’ de JMF (no confundir con la carpeta ‘bin’ de la que acabamos de hablar en el anterior apartado). Por ejemplo, si hemos dejado la ruta por defecto que el instalador de Teletrófono nos ofrecía, la carpeta que deberemos abrir será: C:\Archivos de programa\Teletrófono\bin”. Una vez estemos en esa carpeta deberemos hacer doble clic sobre ‘jmfregistry’. Nos aparecerá una ventana con varias pestañas. Deberemos situarnos en la pestaña llamada “Capture Devices”, pulsar el botón “Detect Capture Devices” de esta pestaña y a continuación pulsar el botón “Commit”. Este paso es muy importante porque hará que JMF detecte los dispositivos de captura de nuestro ordenador, para que así podamos usarlos en Teletrófono.

\*No podemos obviar este último paso aunque previamente a ejecutarlo ya aparezcan dispositivos de captura; es muy probable que los dispositivos de captura que aparecen no sean los de nuestra máquina sino los de la máquina con la que se creó el instalador.

Ahora ya tendremos Teletrófono instalado y configurado en nuestra máquina. Para comenzar a usarlo tan sólo tendremos que buscarlo en el menú ‘Inicio’ de nuestro ordenador.

**\*Nota 2:** Para instalar Java y JMF podemos acudir a las siguientes páginas web:

- Para instalar *Java Development Kit* (Java JDK) podemos acudir a la página <http://java.sun.com/javase/downloads/widget/jdk6.jsp>, donde aparte de poder descargar el ejecutable podemos acceder a las instrucciones de instalación de éste y otros documentos de ayuda.

- Para instalar JMF podemos acudir a la página web <http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/download.html>, donde encontramos tanto el ejecutable como la posibilidad de acceder a documentos de ayuda para la instalación.



## Herramientas de Teletrófono

Cuando accedemos a Teletrófono nos aparece una pantalla con el menú principal de la aplicación. Desde esta pantalla podemos acceder a las diferentes herramientas que podemos utilizar. En concreto, tal y como vemos en la figura I.1 nos aparecen cuatro “cuadros” colgados en nuestra “sala de espera”.



Figura I.1. Menú principal de Teletrófono.

Cada vez que situemos el puntero del ratón sobre alguno de estos “cuadros” aparecerá el nombre de la aplicación que lanzaremos si pulsamos sobre él. En concreto:

- Si situamos el cursor sobre el primer “cuadro” nos aparecerá la frase “Iniciar Videoconferencia desde el extremo Local”. Si pulsamos sobre él, lanzaremos la herramienta de Videoconferencia que debemos utilizar en el lado local, es decir, en el aula del centro educativo en la que se encuentran los alumnos.
- Si situamos el cursor sobre el segundo “cuadro” nos aparecerá la frase “Iniciar Videoconferencia desde el lado Remoto”. Si pulsamos sobre él, lanzaremos la herramienta de Videoconferencia que debemos utilizar en el lado remoto, es decir, en el lugar de interés hasta donde se ha desplazado un profesor.
- Si situamos el cursor sobre el tercer “cuadro” nos aparecerá la frase “Reproducir Grabación”. Si pulsamos sobre él, lanzaremos el Reproductor Teletrófono.
- Si situamos el cursor sobre el último de los “cuadros” nos aparecerá la frase “Editar y Re-sincronizar Subtítulos”. Si pulsamos sobre él, lanzaremos el Editor de Subtítulos Teletrófono.

En la figura I.2 podemos ver lo que sucede si, por ejemplo, nos situamos sobre el “cuadro” central de la pantalla del menú principal.



Figura I.2. Menú principal de Teletrófono: “Iniciar Videoconferencia desde el extremo Remoto”.



*Si en algún momento mientras trabajamos con Teletrófono no se cargan bien las imágenes del menú principal, o cualquier otra imagen, se recomienda se amplíe la memoria de la máquina virtual de Java como de indica a continuación:*

*Lo único que hay que hacer es abrir el panel de control del ordenador, hacer doble clic en ‘Java’, en la ventana que nos aparece seleccionar la pestaña Java y pulsar el botón que permite ver la configuración del ‘Java Runtime Enviroment’. Aparecerá entonces una nueva ventana que muestra las versiones de Java instaladas; en la fila de aquella versión que nos interese, nos dirigimos a la columna “Parámetros del entorno...” y escribimos ‘-Xmx256m’. Por último pulsamos aceptar. Con esta acción habremos doblado el tamaño máximo de la memoria de tipo ‘heap’ de la máquina virtual de Java.*

## La Videoconferencia con Teletrófono

Antes de comenzar a explicar cómo llevar a cabo una videoconferencia con Teletrófono debemos aclarar el significado de dos términos que van a aparecer continuamente a lo largo de este manual: “extremo local” y “extremo remoto”.

Hemos de tener claro antes de comenzar a usar Teletrófono, que la herramienta ha sido especialmente diseñada para el ámbito educativo. Estudiando las necesidades en este tipo de entorno, se han implementado distintas funcionalidades para cada lado de la comunicación. Tendremos por tanto un lado de la comunicación al que llamaremos “extremo local” y otro lado de la comunicación al que llamaremos “extremo remoto”.

Habr  que tener entonces claro qu  papel concreto juega cada uno de los lados de la comunicaci n para saber qu  parte de la aplicaci n Teletr fono se debe ejecutar. El “extremo local” deber  pulsar el icono “Iniciar Videoconferencia desde Local”, y el “extremo remoto” deber  pulsar el icono “Iniciar Videoconferencia desde Remoto”. Llevar a cabo la asignaci n de papeles para ver qu  lado es el extremo local y cu l el remoto es muy sencillo.

El lado local de la comunicaci n estar  situado en el aula del centro educativo donde est n los alumnos, normalmente acomp  ados de un profesor. Este lado de la comunicaci n est  a la espera de poder establecer comunicaci n con el otro extremo para poder recibir la imagen y voz del profesor que se ha trasladado a otro lugar.

El lado remoto de la comunicaci n ser  el lugar de inter s (fuera del centro educativo) que un profesor a escogido mostrar a sus alumnos. Este lado de la comunicaci n ser  el que deba transmitir informaci n audiovisual al otro extremo.



*El “extremo local” de la videoconferencia es el aula donde se encuentran los alumnos a la espera de informaci n audiovisual. El “extremo remoto” de la videoconferencia es el lugar, fuera del centro educativo, hasta donde se ha desplazado un profesor que quiere transmitir informaci n audiovisual a los alumnos que est n en el aula.*

### **Videoconferencia desde el extremo Local**



Si estamos situados en el lado local de la comunicaci n, es decir, en el aula con los alumnos, y queremos iniciar una videoconferencia, deberemos pulsar el primero de los iconos que aparece en la ventana de men  principal de Teletr fono en el que se indica “Iniciar Videoconferencia desde el extremo Local”. Entonces nos aparecer  una ventana como la que se muestra en la figura I.3.

Es la ventana de configuraci n de la videoconferencia y, a trav s de ella, proporcionaremos a Teletr fono los datos que necesita para conectar con el otro extremo.

En primer lugar, la ventana nos pide el “T tulo de la videoconferencia”, aunque no es obligatorio introducir un t tulo.

Despu s nos aparece un cuadro de texto con la direcci n IP de nuestro ordenador y un cuadro de texto vac o en el que debemos introducir la direcci n IP del ordenador con el que deseamos contactar a trav s de la videoconferencia. Para conocer la direcci n IP del otro extremo deberemos comunicarnos con  l, antes de iniciar la videoconferencia, a trav s de un medio de comunicaci n alternativo, por ejemplo, por tel fono. En esa

conversación previa al inicio de la videoconferencia deberá haber un intercambio de direcciones IP.

Figura I.3. Ventana de configuración de la videoconferencia desde el extremo local.



*Una dirección IP, es como un DNI para máquinas, es un número único que utilizan los dispositivos para identificarse y comunicarse entre ellos en una red que utiliza el Internet Protocol (protocolo de Internet). Los números usados en las direcciones IP van desde 0.0.0.0 a 255.255.255.255, aunque algunos de estos valores están reservados para propósitos específicos.*

A continuación, la ventana de configuración de la videoconferencia nos muestra dos cuadros de texto que podemos desplegar si pulsamos sobre ellos. En el primer cuadro nos aparecerá una lista con los dispositivos de captura de audio, existentes en nuestro ordenador, que podemos escoger. Debemos pulsar sobre el nombre del dispositivo que queramos que capture nuestro audio; el audio que capture ese dispositivo será el que durante la videoconferencia se envíe al otro lado de la comunicación. Si pulsamos sobre el segundo cuadro, nos aparecerá una lista con los formatos de audio en los que el dispositivo que acabamos de escoger puede capturar. Si tenemos dudas sobre qué dispositivo o formato escoger, siempre podemos dejar aquellos que aparecen por defecto.

Por último, en la ventana de configuración, aparecen los valores de los puertos que se van a utilizar en cada extremo para llevar a cabo la comunicación. Aparecen ocho valores. La primera columna muestra los puertos que se van a utilizar en este extremo (local) y la segunda columna muestra los puertos que se van a utilizar en el extremo opuesto (remoto). Los valores que se configuren en esta pantalla deberán ser coherentes con los valores que se configuren en el lado remoto de la comunicación, por ello se recomienda dejar los valores por defecto.



*Se recomienda efusivamente que se dejen los valores de los puertos que aparecen por defecto en la ventana de configuración de la videoconferencia.*



Si de todos modos se desea modificar estos valores, deberá pulsar antes el botón “Modificar Puertos”.



Una vez hayamos introducido los datos de configuración, debemos pulsar el botón “Confirmar Configuración”. Entonces, si los datos de configuración son correctos, aparecerá la ventana de videoconferencia que se muestra en la figura I.4.



Figura I.4. Ventana de inicio de videoconferencia del extremo local.



En esta ventana nos aparece un gran aviso en el que se nos indica que para iniciar la videoconferencia debemos pulsar el botón “Iniciar videoconferencia”.



Además, en la parte inferior de la pantalla, nos aparece el botón “Mostrar panel de inserción de subtítulos”. Este botón mostrará el panel a través del cual podremos insertar rótulos bajo la imagen que recibamos desde el extremo remoto una vez hayamos iniciado la videoconferencia.

Iniciemos entonces la videoconferencia pulsando el botón “Iniciar Videoconferencia”. Entonces, tras unos segundos de espera, y siempre que el usuario en el otro extremo también haya pulsado su botón de inicio de la videoconferencia, nos aparecerá la imagen del profesor en remoto por pantalla así como podremos mantener una



conversación con él. En la figura I.5 podemos ver cómo queda nuestra pantalla de videoconferencia una vez se haya establecido la comunicación.



Figura I.5. Videoconferencia Teletrófono desde el lado local.



Como podemos ver en la anterior figura, ha aparecido el botón “Finalizar videoconferencia”, que nos permite finalizar la comunicación con el otro extremo cuando deseemos.



Probemos ahora a pulsar el botón que nos muestra el panel de inserción de subtítulos. Como vemos en la imagen I.6 lo que ocurre es que aparece un cuadro de texto vacío y un botón con un símbolo ‘+’ a su derecha.



Figura I.6. Panel de inserción de subtítulos.

Lo que nos permite este panel es insertar bajo la imagen que recibimos del otro extremo un subtítulo cualquiera. Por ejemplo, vamos a escribir en el cuadro de texto la siguiente frase “La profesora de Ciencias Naturales Anabel Gómez se ha desplazado hasta el parque del Retiro”.



Figura I.7. Introduciendo un subtítulo.



A continuación pulsamos el botón que aparece a la derecha del cuadro de texto e inmediatamente aparecerá el texto insertado bajo la imagen que estamos recibiendo desde el lado remoto.



Si después pulsamos el botón que oculta el panel de subtítulos nos quedaría la ventana de videoconferencia tal y como se muestra en la figura I.8.



*Cada subtítulo podrá tener una longitud máxima de ciento veinticinco caracteres.*



Figura I.8. Ventana de videoconferencia desde el lado local con introducción de subtítulos.

Con las indicaciones seguidas hasta ahora, hemos sido capaces de establecer una comunicación de audio con el extremo remoto así como hemos logrado ver la imagen del profesor que se encuentra en ese extremo. También hemos aprendido a usar la herramienta que nos permite introducir subtítulos, como hemos visto, de una forma muy

sencilla. Pero ahora nos queda por aprender a usar otra de las funcionalidades que nos ofrece la videoconferencia Teletrófono: la grabación de la sesión.



Para comenzar a grabar la sesión de videoconferencia deberemos pulsar el botón “Comenzar grabación”. En cuanto pulsemos este botón, nos aparecerá una ventana a través de la cual deberemos seleccionar la ruta en la que deseamos guardar la grabación, así como el nombre del archivo, como vemos en la siguiente figura:

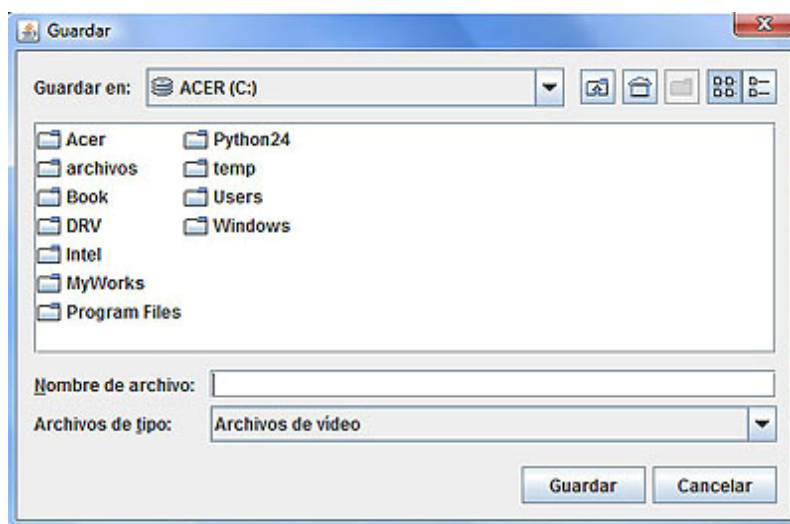


Figura I.9. Ventana de selección de la ruta y nombre del archivo de vídeo.

En cuanto hayamos seleccionado la ruta y escrito el nombre que queremos darle a la grabación, pulsaremos el botón “Guardar”. Inmediatamente comenzará a almacenarse en la ruta escogida la imagen y voz que recibimos del profesor en remoto, así como los rótulos que se inserten desde este momento.



*Quando optamos por grabar la sesión de videoconferencia, lo que hace internamente la aplicación es crear un archivo de vídeo de tipo AVI (con la extensión ‘.avi’) que contendrá la imagen que estamos recibiendo desde el otro extremo, un archivo de audio de tipo GSM (con la extensión ‘.gsm’) que contendrá el audio que estamos recibiendo desde el otro extremo. Además, si se han insertado subtítulos desde que se comenzó a grabar la sesión, se almacenará también un archivo (con la extensión ‘.srt’), que contendrá los rótulos que se han insertado mientras se grababa la sesión.*



*Sólo se puede realizar una grabación por sesión de videoconferencia.*



*Se pueden grabar un máximo de cincuenta subtítulos por sesión de videoconferencia.*





Cuando queramos finalizar la grabación de la sesión de videoconferencia deberemos pulsar el botón “Parar grabación”. Habrá que esperar un instante hasta que el programa realice las operaciones necesarias. Entonces nos aparecerá un mensaje por pantalla indicándonos que la grabación se ha guardado con éxito.

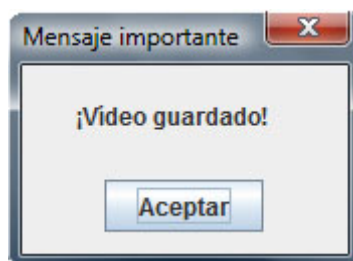


Figura I.10. Mensaje que aparece al finalizar la grabación de la videoconferencia.

Una vez hayamos finalizado la grabación podremos, bien seguir con la videoconferencia, o bien pulsar el botón “Finalizar videoconferencia”, que finalizará la comunicación entre ambos extremos y nos llevará de nuevo a la ventana del menú principal de Teletrófono.



*Si se ha iniciado la grabación de la sesión, la aplicación no nos permitirá finalizar la videoconferencia a menos que previamente hayamos finalizado la grabación.*

### **Videoconferencia desde el extremo Remoto**



Si estamos situados en el lado remoto de la comunicación, es decir, somos el profesor que se ha trasladado a un determinado lugar de interés, y queremos iniciar una videoconferencia, deberemos pulsar el icono central que aparece en la ventana de menú principal de Teletrófono en el que se indica “Iniciar Videoconferencia desde el extremo Remoto”. Entonces nos aparecerá una ventana como la que se muestra en la figura I.11.

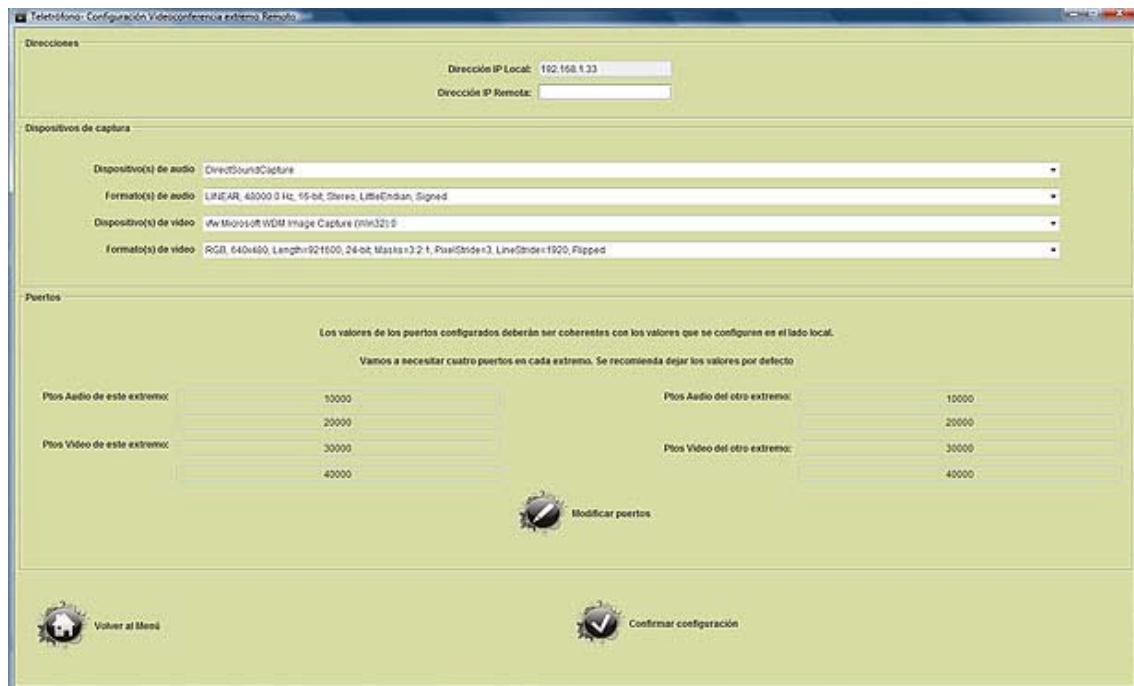


Figura I.11. Ventana de configuración de la videoconferencia desde el extremo remoto.

Es la ventana de configuración de la videoconferencia y, como vemos, es muy parecida a la que nos aparecía al configurar la videoconferencia para el extremo local. Veamos qué información nos pide ahora la ventana de configuración del lado remoto.

El primer elemento que aparece es un cuadro de texto con la dirección IP de nuestro ordenador y un cuadro de texto vacío en el que debemos introducir la dirección IP del ordenador con el que deseamos contactar a través de la videoconferencia. Para conocer la dirección IP del otro extremo deberemos comunicarnos con él, antes de iniciar la videoconferencia, a través de un medio de comunicación alternativo, por ejemplo, por teléfono. En esa conversación previa al inicio de la videoconferencia deberá haber un intercambio de direcciones IP.

A continuación, la ventana de configuración de la videoconferencia nos muestra cuatro cuadros de texto que podemos desplegar si pulsamos sobre ellos. En el primer cuadro nos aparecerá una lista con los dispositivos de captura de audio, existentes en nuestro ordenador, que podemos escoger. Debemos pulsar sobre el nombre del dispositivo que queramos que capture nuestro audio; el audio que capture ese dispositivo será el que durante la videoconferencia se envíe al otro lado de la comunicación. Si pulsamos sobre el segundo cuadro, nos aparecerá una lista con los formatos de audio en los que el dispositivo que acabamos de escoger puede capturar. Si pulsamos sobre el tercer cuadro nos aparecerá una lista con los dispositivos de captura de vídeo, existentes en nuestro ordenador, de entre los que deberemos escoger uno; el vídeo que capture ese dispositivo será el que durante la videoconferencia se envíe al otro lado de la comunicación. Por último, si pulsamos sobre el cuarto cuadro de texto, nos aparecerá una lista con los formatos de vídeo en los que el dispositivo de vídeo que acabamos de escoger puede capturar. Si tenemos dudas sobre qué dispositivos o formatos escoger, siempre podemos dejar aquellos que aparecen por defecto.



*Se recomienda que se tenga especial cuidado con el tamaño de imagen escogido para la captura del vídeo. Se recomienda un tamaño grande de aproximadamente 640x480 píxeles.*

Después, en la ventana de configuración, aparecen los valores de los puertos que se van a utilizar en cada extremo para llevar a cabo la comunicación. Aparecen ocho valores. La primera columna muestra los puertos que se van a utilizar en este extremo (remoto) y la segunda columna muestra los puertos que se van a utilizar en el extremo opuesto (local). Los valores que se configuren en esta pantalla deberán ser coherentes con los valores que se configuren en el lado local de la comunicación, por ello se recomienda dejar los valores por defecto.



*Se recomienda efusivamente que se dejen los valores de los puertos que aparecen por defecto en la ventana de configuración de la videoconferencia.*



Si de todos modos se desea modificar estos valores, deberá pulsar antes el botón “Modificar Puertos”.



Una vez hayamos introducido los datos de configuración, debemos pulsar el botón “Confirmar Configuración”. Entonces, si los datos de configuración son correctos, aparecerá la ventana de videoconferencia que se muestra en la figura I.12.



Figura I.12. Ventana de inicio de videoconferencia del extremo remoto.



En esta ventana nos aparece un gran aviso en el que se nos indica que para iniciar la videoconferencia debemos pulsar el botón “Iniciar videoconferencia”. Iniciemos

entonces la videoconferencia pulsando dicho botón. Tras unos segundos de espera, y siempre que el usuario en el otro extremo también haya pulsado su botón de inicio de la videoconferencia, nos aparecerá en pantalla nuestra propia imagen, comenzaremos la transmisión de ésta al extremo local y además comenzará la transmisión y recepción de audio, pudiendo ya mantener una conversación de voz con el otro lado de la comunicación.

En la figura I.13 podemos ver cómo queda nuestra pantalla de videoconferencia una vez se haya establecido la comunicación.

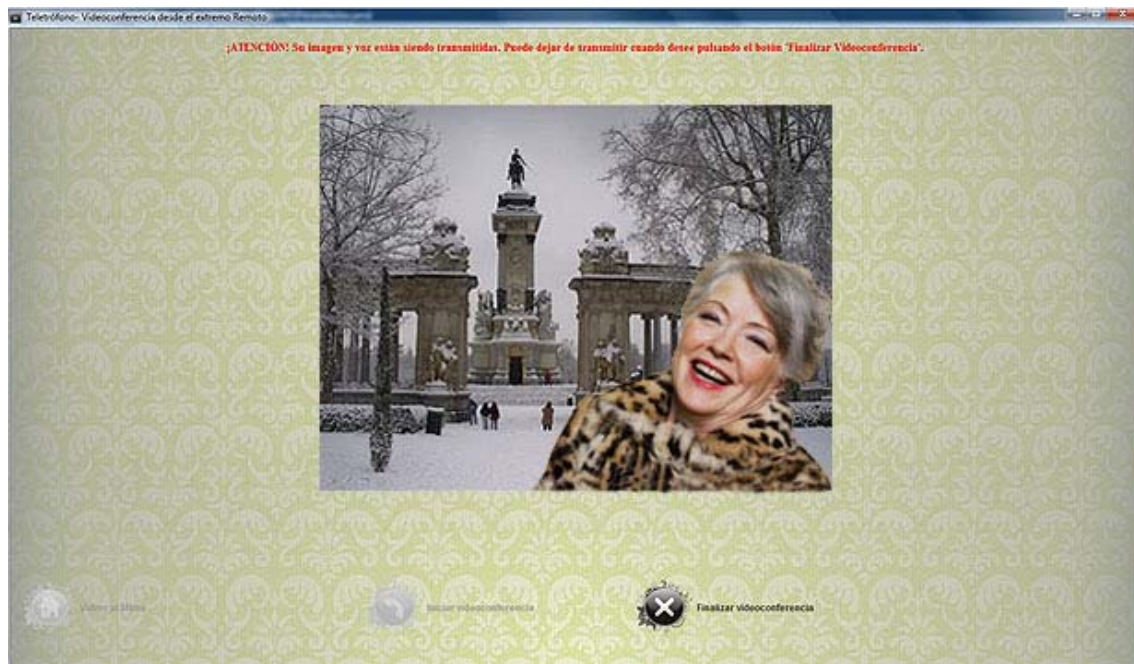


Figura I.13. Videoconferencia Teletrófono desde el lado remoto.

Vemos que la pantalla de videoconferencia en el lado remoto de la comunicación es más sencilla que la que teníamos en el lado local. De este modo el profesor en remoto tendrá menos carga de trabajo y podrá dedicarse de lleno a la explicación.

En la figura I.13 vemos que una vez iniciamos la videoconferencia, lo que visualizamos en la pantalla del lado remoto es nuestra propia imagen. Esto nos servirá de gran ayuda para saber qué enfoque exacto del entorno estamos ofreciendo a distancia a los alumnos en el aula.

El único botón habilitado ahora es el de finalizar la videoconferencia. Si pulsamos dicho botón, dejaremos inmediatamente de transmitir nuestra imagen y voz al otro extremo, se cerrará la ventana de videoconferencia y nos aparecerá de nuevo la ventana de menú principal de Teletrófono.



*Hay que tener especial cuidado cuando desde este extremo pulsamos el botón de iniciar videoconferencia, pues en breves instantes, se comienza a transmitir nuestra imagen al otro extremo.*



*Para evitar sorpresas, en la parte superior de la ventana de videoconferencia desde remoto, aparece siempre un letrero que nos indica si nuestra imagen está siendo transmitida. En la figura I.12 podemos ver que el*



*letrero es pequeño y con letras negras, indicando: “En este momento su imagen no está siendo transmitida”. Sin embargo en la figura I.13 aparece en su lugar un letrero con las letras de color rojo y más grandes que indica: “¡ATENCIÓN! Su imagen y voz están siendo transmitidas. Puede dejar de transmitir cuando desee pulsando el botón ‘Finalizar videoconferencia’”.*



*En el caso en que se deseen realizar varias videoconferencias seguidas se recomienda que, para cada una de ellas, se inicie de nuevo el programa Teletrófono, de otro modo la cámara web podría dejar de capturar correctamente nuestra imagen.*

## El Reproductor Teletrófono



Para ejecutar el Reproductor Teletrófono deberemos pulsar el tercer icono que aparece en la ventana de menú principal en el que se indica “Reproducir Grabación”. Esta herramienta nos va a permitir reproducir todas aquellas sesiones de videoconferencia que hayamos grabado con Teletrófono. El Reproductor mostrará tanto el vídeo con las imágenes y audio de la sesión como los subtítulos.

Cuando pulsamos el icono del menú principal que lanza el Reproductor lo primero que nos aparece es una ventana para seleccionar a través de ella la ruta y nombre del archivo de vídeo que queremos reproducir. Por defecto, esta ventana sólo mostrará los archivos cuya extensión sea ‘.mov’, ‘.avi’ o ‘.mpg’. En la figura I.14 podemos ver el aspecto que tiene la ventana a través de la cual seleccionamos el archivo que queremos reproducir.

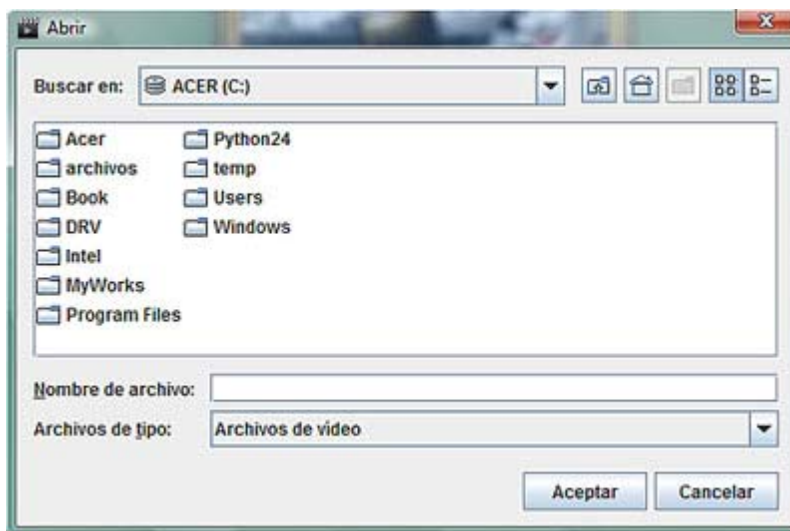


Figura I.14. Selector de archivos del Reproductor Teletrófono.

El Reproductor, aunque está especialmente pensado para grabaciones de videoconferencia realizadas a través de Teletrófono (que sabemos que siempre se guardan en un archivo de vídeo del tipo ‘.avi’, un archivo de audio del tipo ‘.gsm’ y un archivo ‘.srt’, en el caso en que se inserten subtítulos durante la grabación), también permite la reproducción de archivos de vídeo con la extensión ‘.mov’ o ‘.mpg’. Asimismo, permite la reproducción de vídeos en cualquiera de los tres formatos válidos tengan o no un archivo de subtítulos asociado.



*Se considera que un archivo de vídeo tiene subtítulos asociados si existe un archivo SRT (‘.srt’) ubicado en el mismo directorio que éste y cuyo nombre sea idéntico al archivo de vídeo (sin incluir la extensión).*



*Teletrófono incluye un vídeo de prueba en una carpeta llamada “video\_ejemplo\_Teletrofono”. En esta carpeta está la grabación de una sesión de videoconferencia con Teletrófono que incluye: un archivo de vídeo ‘.avi’ (que es el que debemos seleccionar en el selector de archivos del Reproductor), un archivo de audio ‘.gsm’ y un archivo con subtítulos ‘.srt’.*

Cuando hayamos escogido el archivo que queremos reproducir pulsamos en la ventana del selector el botón ‘Aceptar’. Si el archivo escogido es de tipo ‘.mov’, ‘.avi’ o ‘.mpg’, entonces se abrirá la ventana del Reproductor Teletrófono y comenzará la reproducción.

Como vemos en la figura I.15, en el caso en el que el archivo de vídeo seleccionado, tenga subtítulos asociados estos se mostrarán en un cuadro de texto bajo la reproducción del vídeo. Estos subtítulos irán cambiando conforme vaya avanzando la reproducción, de acuerdo con los tiempos en los que fueron insertados durante la videoconferencia. En el caso en que no exista un archivo de subtítulos asociado, no hay problema, el reproductor mostrará por pantalla únicamente el archivo de vídeo escogido y en el cuadro de texto mostrará la frase: “(No hay subtítulos)”.

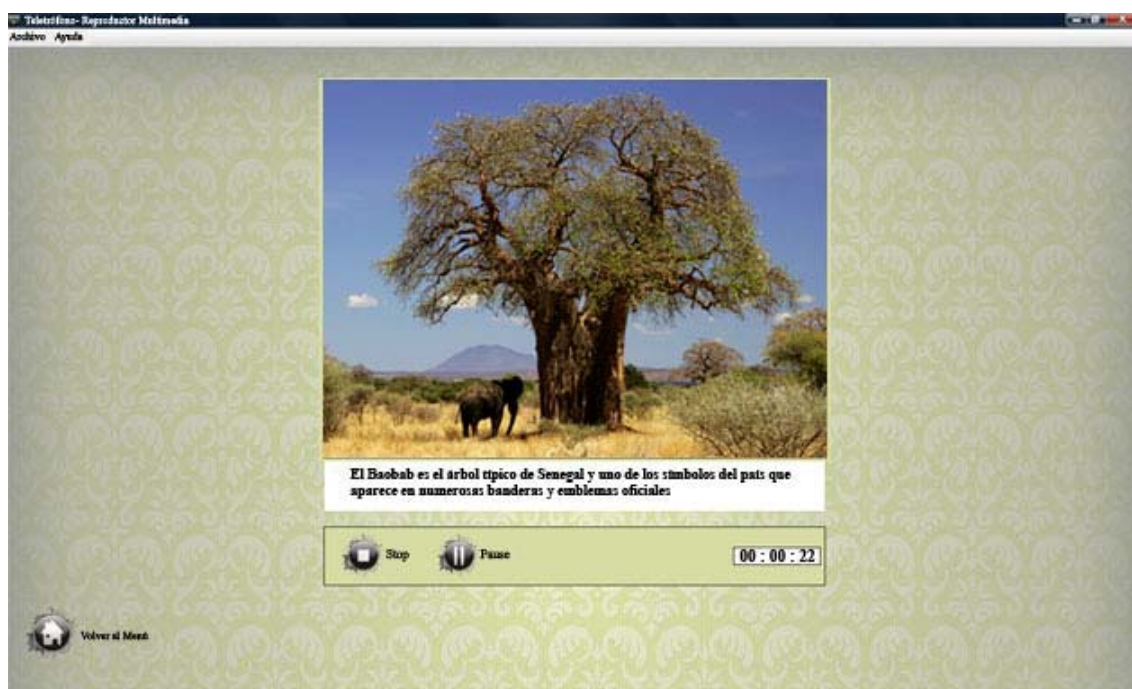


Figura I.15. Ventana del Reproductor Teletrófono.

En la anterior figura podemos observar la pantalla principal del reproductor. En ella aparece uno de los fotogramas del vídeo que está reproduciéndose y, bajo éste, el cuadro de texto donde van apareciendo los subtítulos asociados al vídeo. En el panel de control aparecen dos botones, ahora son los de “Stop” y “Pause”, porque la reproducción está en curso. Si la reproducción está parada tan sólo aparecerá el botón de “Play”. Y si la reproducción está pausada aparecerán los botones de “Stop” y “Resume”.



*Los archivos que contienen los subtítulos están pensados para que se creen, se modifiquen y se reproduzcan mediante Teletrófono, a través de sus herramientas de videoconferencia, el editor y el reproductor, respectivamente. Es por ello aconsejable, que se eviten modificaciones de estos archivos fuera de Teletrófono ya que podrían dar lugar a errores cuando quisiésemos volver a usarlos con nuestro programa.*

### **El Editor de Subtítulos Teletrófono**



Otra de las herramientas que nos ofrece Teletrófono es el Editor de Subtítulos. Se puede acceder a esta herramienta a través del último icono que aparece en el menú principal en el que figura la frase: “Editar y Re-sincronizar Subtítulos”. Esta herramienta nos va a ser de gran utilidad para modificar tanto el contenido como los tiempos de inicio y fin de cada uno de los subtítulos asociados a una grabación de una sesión de videoconferencia.



*Se recomienda, una vez más, que los archivos que contienen los subtítulos de las sesiones de videoconferencia sean manipulados exclusivamente a través de Teletrófono, nunca manualmente. Para ello se ha creado esta herramienta de edición. Evitaremos así errores indeseados.*

Cuando el usuario escoge esta opción, nos aparece una ventana sobre el menú principal. La ventana contiene un cuadro de texto en el que se nos pide que introduzcamos la ruta en la que se encuentra el archivo de vídeo cuyos subtítulos queremos editar. Como el cuadro de texto no nos deja escribir directamente sobre él, debemos pulsar el botón punteado que aparece a su derecha.

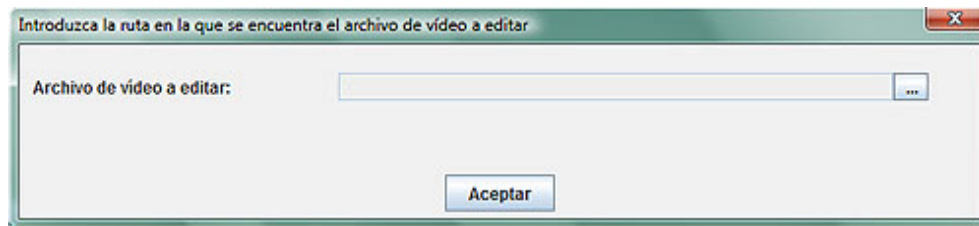



Figura I.16. Diálogo que aparece al iniciar el Editor Teletrófono.

Al pulsar el botón punteado  nos aparecerá en pantalla un selector de archivos idéntico al que mostrábamos en la figura I.14.



*Teletrófono incluye un vídeo de prueba en una carpeta llamada “video\_ejemplo\_Teletrofono”. En esta carpeta está la grabación de una sesión de videoconferencia con Teletrófono que incluye: un archivo de vídeo ‘.avi’ (que es el que debemos seleccionar en el selector de archivos del Editor), un archivo de audio ‘.gsm’ y un archivo con subtítulos ‘.srt’.*

El editor de subtítulos ha sido diseñado especialmente para tratar con vídeos y archivos de subtítulos que hayan sido creados a través de Teletrófono. El Editor, sólo admite entonces archivos de vídeo de tipo AVI (cuya extensión sea ‘.avi’), que lleven un archivo de audio GSM y subtítulos asociados (pues no tendría sentido editar los subtítulos de un vídeo que no tiene subtítulos asociados) y cuyas dimensiones de imagen sean como máximo 352x288 píxeles.

Una vez hayamos escogido el vídeo deseado (que cumpla las anteriores condiciones) y hayamos pulsado el botón ‘Aceptar’ del selector, aparecerá por pantalla la ventana del Editor Teletrófono que se muestra en la figura I.17.



Figura I.17. Ventana del Editor Teletrófono.





*El Editor Teletrófono sólo trabaja con vídeos cuya extensión sea ‘.avi’, que lleven un archivo de audio ‘.gsm’ y otro de subtítulos asociados y cuyas dimensiones de imagen sean como máximo 352x288 píxeles.*

En la parte superior izquierda de la ventana del editor aparece un cuadro con el texto: “Pulse Play para reproducir el vídeo escogido”. Se trata de un reproductor, muy parecido al Reproductor Teletrófono que permite reproducir el vídeo escogido y sus subtítulos.

En la parte central de la pantalla aparece una lista con los subtítulos asociados al vídeo que hemos escogido. En esa lista se reflejan, por orden de aparición, el texto de cada uno de los subtítulos con sus tiempos de inicio y finalización. Cada una de las líneas que aparece en la lista tiene un formato como el de este ejemplo:

00:00:01,000 --> 00:00:10,000 --> ¿Adivináis en qué país estamos?

, el primer tiempo que aparece es el tiempo de inicio del subtítulo, después aparece el tiempo de finalización y por último el texto en sí. Estos son justamente los tres elementos que vamos a poder modificar a través del Editor.



Quando deseemos comprobar cómo quedan esos subtítulos en el vídeo, tan sólo tendremos que pulsar el botón de “Play” del reproductor. Entonces comenzará la reproducción del vídeo y de los subtítulos.



Si queremos modificar el contenido de alguno de los subtítulos o sus tiempos de inicio o fin, deberemos seleccionar dicho subtítulo en la lista (pulsando sobre él), y a continuación pulsar el botón “Editar” (situado bajo la lista). Esto provocará la aparición en pantalla de un nuevo panel: el panel de edición.

En la siguiente figura, la I.18, podemos observar con detalle el panel de edición. Como vemos, en el panel de edición aparecerán los tiempos de inicio y fin del subtítulo así como el propio texto del subtítulo. Todos esos valores podemos modificarlos simplemente situándonos sobre esos campos, borrando los antiguos valores e introduciendo los nuevos.

Tiempo inicio	Tiempo fin	Texto del Subtítulo
00 : 00 : 11	00 : 00 : 15	El profesor Gómez se ha trasladado a Senegal para mostrarnos su país preferido.

Confirmar cambios en el subtítulo

Figura I.18. Panel de edición del Editor Teletrófono.

En la figura I.19 podemos ver el aspecto de la ventana del Editor mientras reproducimos el vídeo con sus subtítulos y mientras editamos uno de estos subtítulos (el subtítulo que está siendo editado aparecerá en la lista subrayado en color azul).



Figura I.19. Ventana del Editor Teletrófono, editando un subtítulo.



*Antes de confirmar los cambios en el subtítulo se recomienda comprobar que se han introducido valores coherentes. Por ejemplo, que el tiempo de inicio del subtítulo sea anterior al tiempo de finalización del mismo, que el tiempo de finalización del subtítulo sea anterior al tiempo en que el vídeo finaliza, que el tiempo de fin del subtítulo sea anterior al tiempo de inicio del siguiente subtítulo, que el tiempo de inicio del subtítulo sea mayor que el tiempo de fin del subtítulo anterior, etc. De todos modos el propio programa comprobará antes de realizar cambio alguno que todas estas condiciones, y otras, se cumplen.*



Una vez hayamos realizado los cambios deseados, deberemos pulsar el botón “Confirmar cambios en el subtítulo”. Entonces el panel de edición desaparecerá y el archivo que contiene los subtítulos se habrá modificado. Apareciéndonos el siguiente mensaje por pantalla:

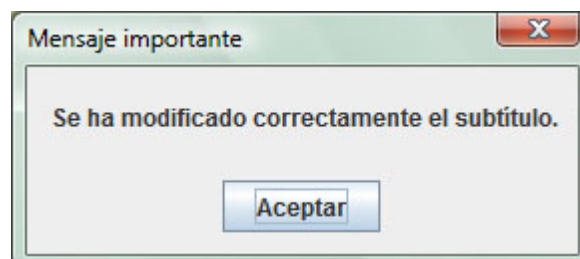


Figura I.20. Mensaje que indica que se ha modificado un subtítulo.

Inmediatamente se actualizará la lista de subtítulos que aparece en el Editor y podremos ver como han quedado los cambios realizados a través del reproductor que contiene el propio Editor.



*Si pulsamos el botón del panel de edición que confirma los cambios en un subtítulo mientras estamos reproduciendo el vídeo, nos aparecerá por pantalla un mensaje informándonos de que para ejecutar los cambios es necesario que el reproductor del vídeo esté parado. El usuario deberá entonces pulsar el botón “Stop” del reproductor e inmediatamente podrá pulsar ya el botón de confirmación de los cambios.*

## **ANEXO II**

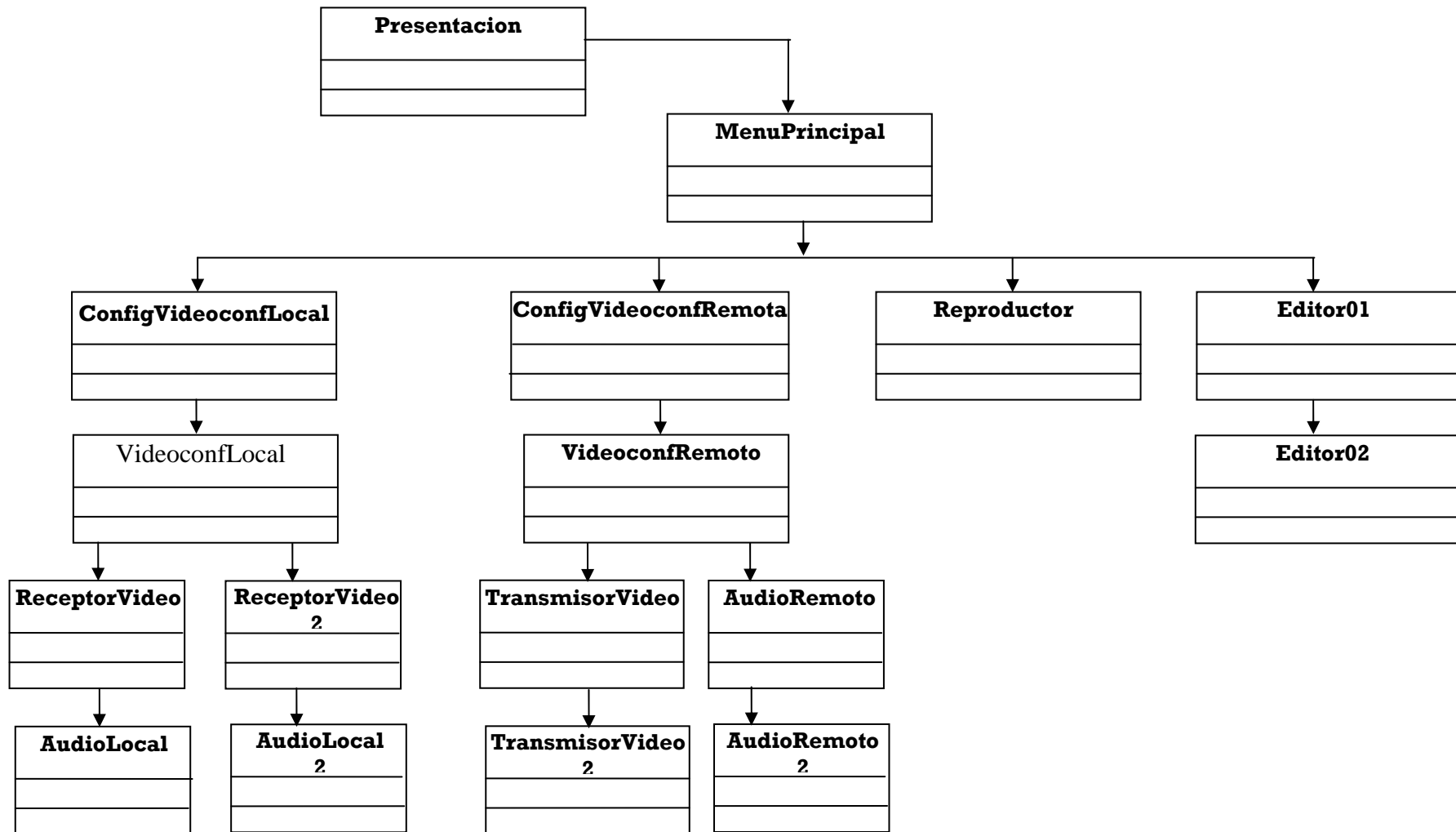
# **DIAGRAMAS DEL CÓDIGO**

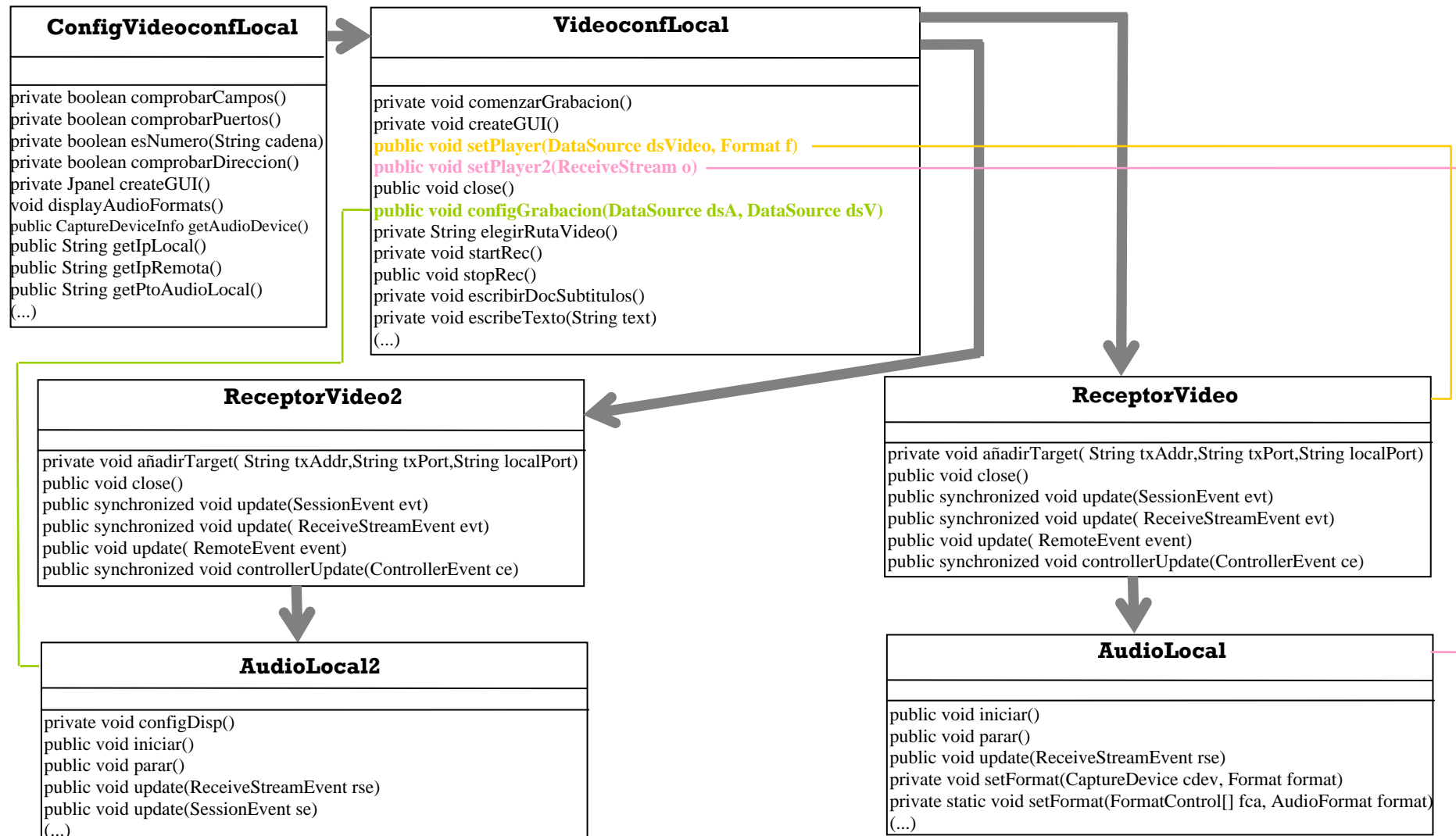
En este anexo se muestran algunos diagramas con las clases usadas en la implementación del software de videoconferencia Teletrófono. Estos esquemas nos ayudarán a comprender cuál es la secuencia de clases que se ejecutan y cómo se interrelacionan entre ellas.

En total, nuestra aplicación se compone de veintisiete clases. En los diagramas incluidos en este anexo tan sólo aparecen las clases más significativas para el funcionamiento de la aplicación.

A continuación se listan las veintisiete clases:

- |                         |                    |                   |
|-------------------------|--------------------|-------------------|
| - Presentacion          | - AudioLocal       | - ChooserAbrir    |
| - MenuPrincipal         | - AudioLocal2      | - ChooserGuardar  |
| - ConfigVideoconfLocal  | - AudioRemoto      | - DialogoAcercaDe |
| - ConfigVideoconfRemota | - AudioRemoto2     | - DialogoAyuda    |
| - Reproductor           | - ReceptorVideo    | - DialogoEstandar |
| - Editor01              | - ReceptorVideo2   | - DialogoSalir    |
| - Editor02              | - TransmisorVideo  | - Boton           |
| - VideoconfLocal        | - TransmisorVideo2 | - FiltroVideo     |
| - VideoconfRemoto       | - Reloj            | - Espere          |

**DIAGRAMA SEGÚN EL ORDEN DE EJECUCIÓN**

**VIDEOCONFERENCIA DESDE EL EXTREMO LOCAL**

**VIDEOCONFERENCIA DESDE EL EXTREMO REMOTO**